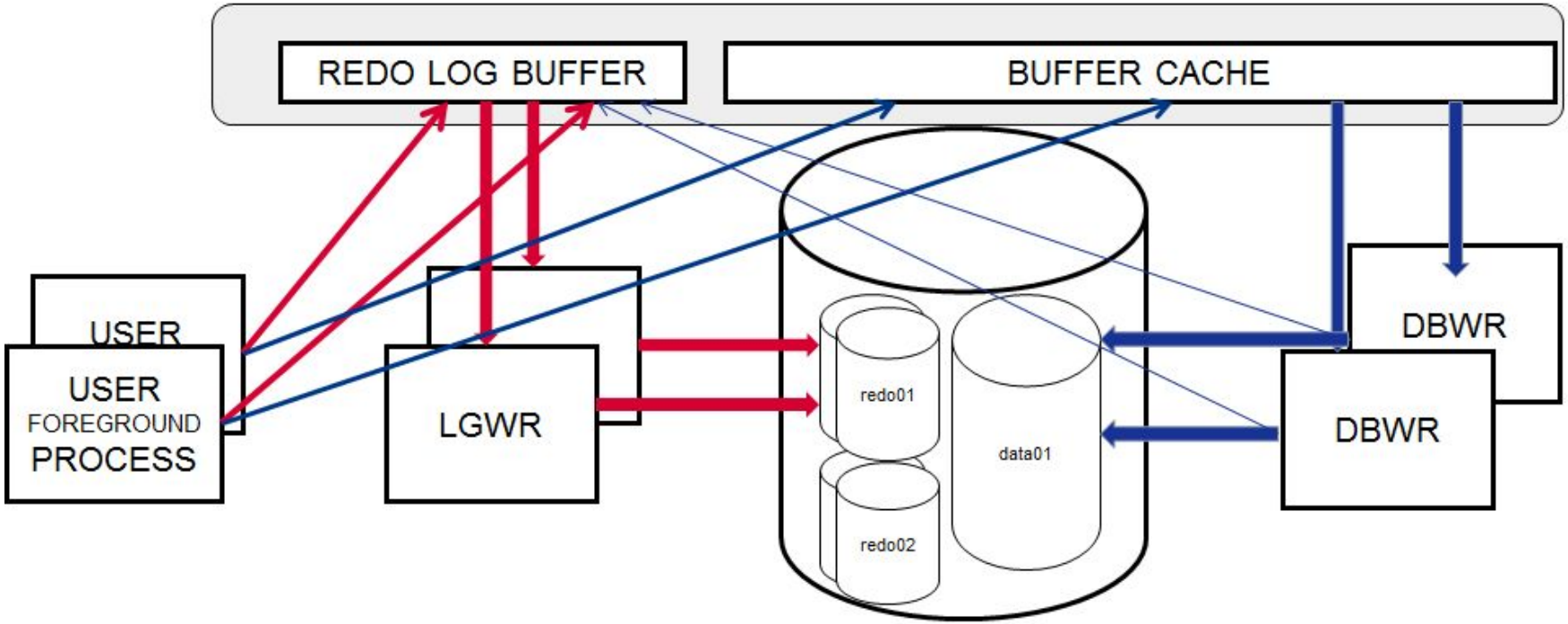
Deutsche Bank
Deutsche Bank Technology Center

Oracle waits “log file sync” and “log file parallel write” on Linux

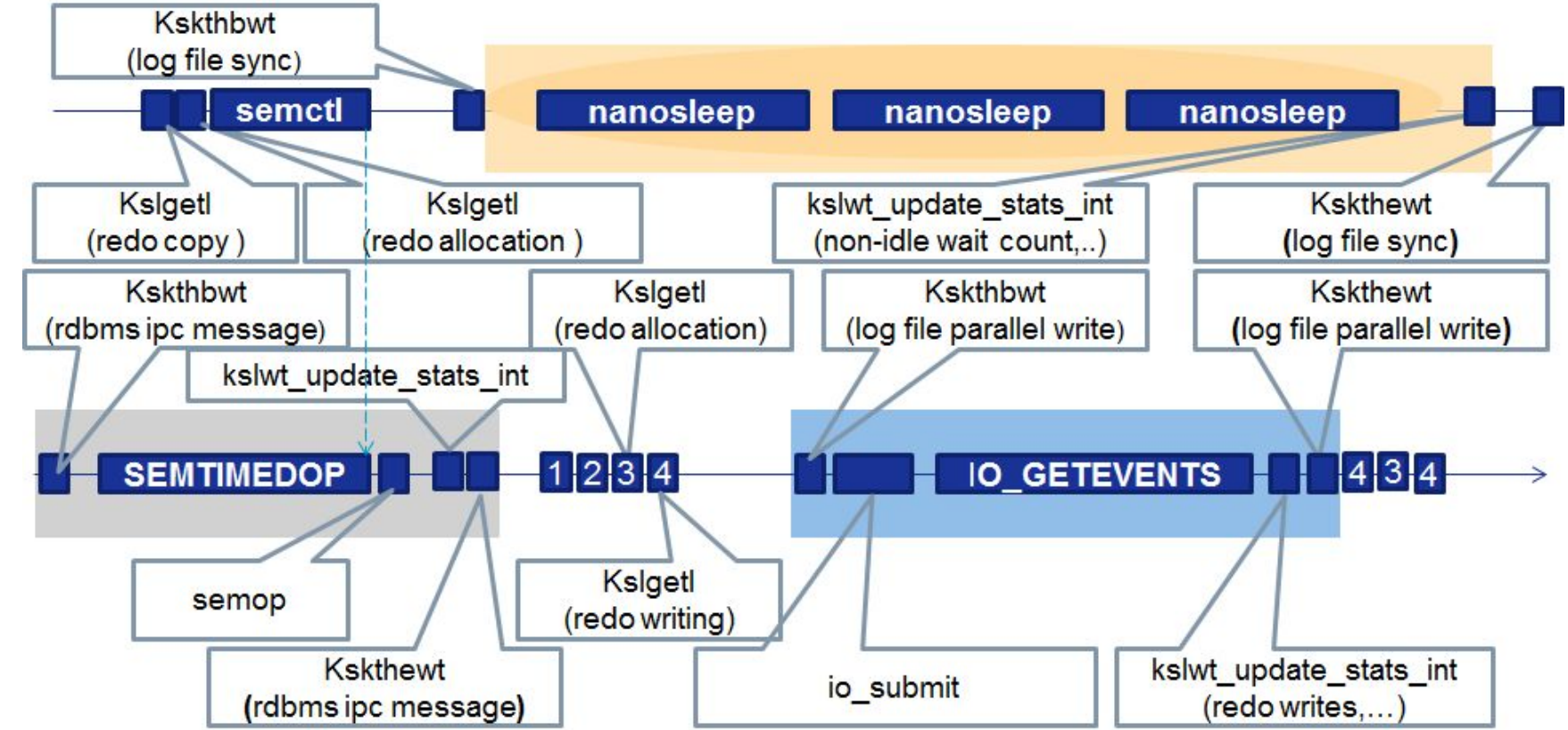
July 2017

Nikolay Kudinov

LGWR Process Overview



You will be able to understand the diagram



Agenda

- Linux semaphores and IO API
  - semget
  - semop, semtimedop,semctl
  - io_submit, io_getevents
- Several oracle internal C functions
- Tools
  - strace
  - systemtap
  - gdb
- LFS and LFPW diagrams

Linux semaphores API - semget

NAME

semget - get a semaphore set identifier

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
int semget(key_t key, int nsems, int semflg);
```

DESCRIPTION

The semget() system call returns the semaphore set identifier associated with the argument key. A new set of nsems semaphores is created if key has the value IPC_PRIVATE or if no existing semaphore set is associated with key and IPC_CREAT is specified in semflg.

If semflg specifies both IPC_CREAT and IPC_EXCL and a semaphore set already exists for key, then semget() fails with errno set to EEXIST. (This is analogous to the effect of the combination O_CREAT | O_EXCL for open(2).)

Upon creation, the least significant 9 bits of the argument semflg define the permissions (for owner, group and others) for the semaphore set.

The argument nsems can be 0 (a don't care) when a semaphore set is not being created. Otherwise nsems must be greater than 0 and less than or equal to the maximum number of semaphores per semaphore set.

RETURN VALUE

If successful, the return value will be the semaphore set identifier (a nonnegative integer), otherwise -1 is returned, with errno indicating the error.

Example (semget)

```
#include <sys/sem.h>
#include <stdio.h>

#define KEY (1490)
int main()
{
  int id;
  id = semget(KEY, 1, 0666 | IPC_CREAT);
  if(id < 0){
    fprintf(stderr, "Unable to obtain semaphore.\n");
  }
  else{
    fprintf(stderr, "Semaphore %d initialized.\n", KEY);
  }
}
```

$strace output:

…
semget(0x5d2, 1, IPC_CREAT|0666) = 524291
write(2, " Semaphore 1490 initialized.\n", 29 Semaphore 1490 initialized.) = 29
…

$ipcs -s -i 524291

Semaphore Array semid=524291
uid=500 gid=500 cuid=500 cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Not set
ctime = Thu **********

semnum	value	ncount	zcount	pid
0	0	0	0	0

semop, semtimedop

NAME

semop, semtimedop - semaphore operations

SYNOPSIS

```
int semop(int semid, struct sembuf *sops, unsigned nsops);
int semtimedop(int semid, struct sembuf *sops, unsigned nsops, struct timespec *timeout);
```

DESCRIPTION

Each semaphore in a semaphore set has the following associated values:

```
unsigned short  semval;   /* semaphore value */
unsigned short  semzcnt;  /* # waiting for zero */
unsigned short  semncnt;  /* # waiting for increase */
pid_t           sempid;   /* process that did last op */
```

semop() performs operations on selected semaphores in the set indicated by semid. Each of the nsops elements in the array pointed to by sops specifies an operation to be performed on a single semaphore. The elements of this structure are of type struct sembuf, containing the following members:

unsigned short sem_num; /* semaphore number */ , short sem_op; /* semaphore operation */ , short sem_flg; /* operation flags */

If sem_op is a positive integer, the operation **adds** this value to the semaphore value (semval)

If sem_op is less than zero, the process must have alter permission on the semaphore set. If semval is greater than or equal to the absolute value of sem_op, the operation can proceed immediately: the absolute value of sem_op is **subtracted** from semval

Example (semop)

```
#define KEY (1490)
void main(){
  int id;
  struct sembuf operations[1];
  int retval;
  id = semget(KEY, 1, 0666);
  if(id < 0){
    fprintf(stderr, "Cannot find semaphore,
     exiting.\n");
    exit(0);
  }
  printf("Process id is %d\n", getpid());
  operations[0].sem_num = 0;
  operations[0].sem_op  = 3;
  operations[0].sem_flg = 0;
  retval = semop(id, operations, 1);
}
```

$strace output:

```
semget(0x5d2, 1, 0666)                  = 524291
getpid()                                = 12024
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7f0b000
write(1, "Process id is 12024\n", 20Process id is 12024
)   = 20
semop(524291, 0xbfef4f06, 1)            = 0
```

$ipcs -s -i 524291

```
Semaphore Array semid=524291
uid=500  gid=500        cuid=500        cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Thu ************************
ctime = Thu ************************
semnum     value      ncount     zcount     pid
0          3          0          0          12024
```

Example (semop)

set operations[0].sem_op = -1

$ipcs -s -i 524291
Semaphore Array semid=524291
uid=500 gid=500 cuid=500 cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Thu **********
ctime = Thu **********
semnum value ncount zcount pid
0 2 0 0 12148

$ipcs -s -i 524291
Semaphore Array semid=524291
uid=500 gid=500 cuid=500 cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Thu **********
ctime = Thu **********
semnum value ncount zcount pid
0 1 0 0 12165

$ipcs -s -i 524291
Semaphore Array semid=524291
uid=500 gid=500 cuid=500 cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Thu **********
ctime = Thu **********
semnum value ncount zcount pid
0 0 0 0 12181

hungs

semget(0x5d2, 1, 0666) = 524291
getpid() = 12186
……
semop(524291, 0xbfa4c356, 1

$ ipcs -s -i524291
……..
otime = Thu **********
ctime = Thu **********
semnum value ncount zcount pid
0 0 1 0 12181

Example (semtimedop)

```
void main()
{
   int id;
   struct sembuf operations[1];
   int retval;
   id = semget(KEY, 1, 0666);
   if(id < 0){
     fprintf(stderr, "Cannot find semaphore, exiting.\n");
     exit(0);
   }
   printf("Process id is %d\n", getpid());
   operations[0].sem_num = 0;
   operations[0].sem_op  = -1;
   operations[0].sem_flg = 0;
   struct timespec timeout = { 10, 0 };
   retval = semtimedop(id, operations, 1, &timeout);
}
```

```
$strace –T
semget(0x5d2, 1, 0666)      = 524291 <0.000009>
getpid()                    = 12445 <0.000008>
……..
semtimedop(524291, 0xbfb5f2f6, 1, {10, 0}) = -1 EAGAIN (Resource temporarily unavailable) <10.001245>
```

semctl

NAME

semctl - semaphore control operations

SYNOPSIS

int semctl(int semid, int semnum, int cmd, ...);

DESCRIPTION

semctl() performs the control operation specified by cmd on the semaphore set identified by semid, or on the semnum-th semaphore of that set. (The semaphores in a set are numbered starting at 0.) This function has three or four arguments, depending on cmd. When there are four, the fourth has the type union semun. The calling program must define this union as follows:

```
union semun {
    int              val;    /* Value for SETVAL */
    struct semid_ds *buf;    /* Buffer for IPC_STAT, IPC_SET */
    unsigned short  *array;  /* Array for GETALL, SETALL */
    struct seminfo  *__buf;  /* Buffer for IPC_INFO   (Linux specific) */
};
```

Valid values for cmd are

……

SETVAL Set the value of semval to arg.val for the semnum-th semaphore of the set, updating also the sem_ctime member of the semid_ds structure associated with the set. Undo entries are cleared for altered semaphores in all processes. If the changes to semaphore values would permit blocked semop() calls in other processes to proceed, then those processes are woken up. The calling process must have alter permission on the semaphore set.

Example (semctl)

```
#define KEY (1490)
int main() {
  int id;
  int ret;
  union semun { int val;struct semid_ds *buf;ushort * array;} argument;
  argument.val = 10;
  id = semget(KEY, 1, 0666 | IPC_CREAT);
  if(id < 0)
  {
    fprintf(stderr, "Unable to obtain semaphore.\n");
    exit(0);
  }
  ret =semctl(id, 0, SETVAL, argument);
}
```

```
ipcs -s -i 524291

Semaphore Array semid=524291
uid=500  gid=500        cuid=500        cgid=500
mode=0666, access_perms=0666
nsems = 1
otime = Thu ************************
ctime = Thu ************************
semnum     value      ncount     zcount     pid
0          10         0          0          13453
```

$strace output:

```
...
semget(0x5d2, 1, IPC_CREAT|0666)       = 524291
semctl(524291, 0, IPC_64|SETVAL, 0xbfc0e8d8) = 0
...
```

ipc

NAME
ipc - System V IPC system calls

SYNOPSIS
int ipc(unsigned int call, int first, int second, int third, void *ptr, long fifth);
DESCRIPTION
ipc() is a common kernel entry point for the System V IPC calls for messages, semaphores, and
shared memory. call determines which IPC function to invoke; the other arguments are passed
through to the appropriate call.

User programs should call the appropriate functions by their usual names. Only standard library
implementors and kernel hackers need to know about ipc().

ipc source code

```
/*
 * sys_ipc() is the de-multiplexer for the SysV IPC calls..
 *
 * This is really horribly ugly.
 */
asmlinkage int sys_ipc (uint call, int first, int second, int third, void __user *ptr, long fifth)
{
        int version, ret;
        version = call >> 16; /* hack for backward compatibility */
        call &= 0xffff;

        switch (call) {
        case SEMOP:
                return sys_semtimedop (first, (struct sembuf __user *)ptr, second, NULL);
        case SEMTIMEDOP:
                return sys_semtimedop(first, (struct sembuf __user *)ptr, second,
                                        (const struct timespec __user *)fifth);
        case SEMGET:
                return sys_semget (first, second, third);
        case SEMCTL: {
                union semun fourth;
                if (!ptr)
                        return -EINVAL;
                if (get_user(fourth.__pad, (void __user * __user *) ptr))
                        return -EFAULT;
                return sys_semctl (first, second, third, fourth);
```

Linux I/O (io_submit)

NAME

io_submit - Submit asynchronous I/O blocks for processing

SYNOPSIS

```
#include <libaio.h>

long io_submit (aio_context_t ctx_id, long nr, struct iocb **iocbpp);
```

DESCRIPTION

io_submit() queues nr I/O request blocks for processing in the AIO context ctx_id. iocbpp should be an array of nr AIO request blocks, which will be submitted to context ctx_id.

RETURN VALUE

io_submit() returns the number of iocbs submitted and 0 if nr is zero.

Linux I/O (io_getevents)

NAME

io_getevents - Read asynchronous I/O events from the completion queue

SYNOPSIS

```
#include <linux/time.h>
#include <libaio.h>
long io_getevents (aio_context_t ctx_id, long min_nr, long nr, struct io_event *events,
                   struct timespec *timeout);
```

DESCRIPTION

io_getevents() attempts to read at least min_nr events and up to nr events from the completion queue of the AIO context specified by ctx_id. timeout specifies the amount of time to wait for events, where a NULL timeout waits until at least min_nr events have been seen. Note that time-out is relative and will be updated if not NULL and the operation blocks.

RETURN VALUE

io_getevents() returns the number of events read: 0 if no events are available or < min_nr if the timeout has elapsed.

Example

```
 char log_buffer1[2097152];
 char log_buffer2[2097152];
 …
 cb1[0].aio_fildes = fd;
 cb1[0].aio_lio_opcode = IO_CMD_PWRITE;
 cb1[0].aio_reqprio = 0;
 cb1[0].u.c.buf = log_buffer1;
 cb1[0].u.c.nbytes = 2097152;
 cb1[0].u.c.offset = 0;

 cb1[1].aio_fildes = fd;
 cb1[1].aio_lio_opcode = IO_CMD_PWRITE;
 cb1[1].aio_reqprio = 0;
 cb1[1].u.c.buf = log_buffer2;
 cb1[1].u.c.nbytes = 2097152;
 cb1[1].u.c.offset = 2097152;

 iocbs1[0] =  &cb1[0];
 iocbs1[1] =  &cb1[1];

 printf("log_buffer1=%p \n",&log_buffer1[0]);
 printf("log_buffer2=%p \n",&log_buffer2[0]);

 res = io_submit(ctx, 2, iocbs1);
……..
 res = io_getevents(ctx, 2, 2, events, NULL);
```

./a.out

……..

io_setup OK ctx=**0x7ffcf1037000**

My process ID : 21581

log_buffer1=**0x7fff103b06a0**

log_buffer2=**0x7fff101b06a0**

io_submit OK: iocbs=0x7fff105b06e0

……..

Systemtap output

……..

kernel.function("sys_io_submit@fs/aio.c:1837").call(+)a.out ctx_id=**0x7ffcf1037000** nr=2

addr=**0x7fff103b06a0** addr2=140733465691808

fd=3 file=test_file.txt op_code=1 offset= 0 **bytes=2097152**

addr=**0x7fff101b06a0** addr2=140733463594656

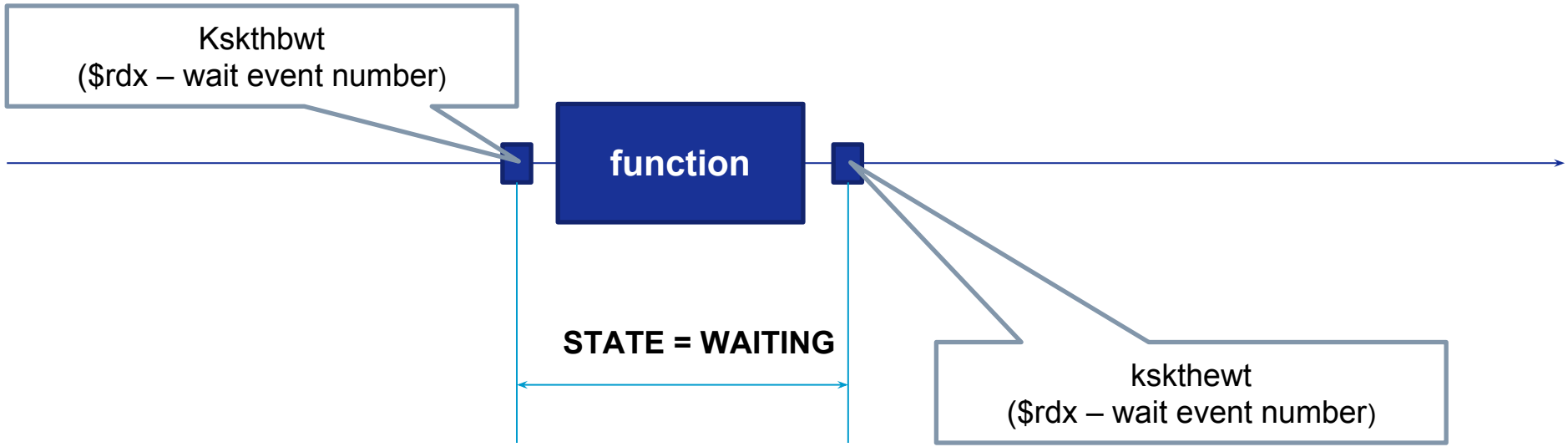
fd=3 file=test_file.txt op_code=1 offset= 2097152 **bytes=2097152**

Kernel Parameter "aio-max-size" does not exist in RHEL4 / EL4 / RHEL5 /EL5 (Doc ID 549075.1)

With the introduction of Asynchronous I/O in RHEL2.1 / 3, (2.4 kernels) a new kernel parameter aio-max-size was added. It can be adjusted to get better I/O throughout in some environments, such as data store/warehouse etc. However this parameter is removed in the Linux 2.6 kernels.

kskthbwt, kskthewt

kskthbwt- Kernel service Kompile thread begin wait
$rdx – wait event number
kskthewt- Kernel service Kompile thread end wait.



kslgetl

kslgetl(laddr, wait, why, where) – get exclusive latch

laddr - address of latch
wait - flag for no-wait (0) or wait (1) latch acquisition
why - integer code for location from where the latch is acquired.
where- integer code ….

kslgetl example

USER process
…….
…….

(gdb) call **kslgetl**(0x**6000e660**,1, 1,1)

LGWR trace file
…….
Breakpoint 2, 0x00007fc20bb9d0c0 in **semtimedop** () from /lib64/libc.so.6
semid=163840
{sem_num=17, sem_op= -1, sem_flg= 0}
nsops=1

Breakpoint 1, 0x000000000c8d9590 in **kslgetl** ()
kslgetl addr=**6000e660**, wait=1, where=0, why=223

Breakpoint 6, 0x000000000c8da6d0 in **kslwtbctx** ()
kslwtbctx event#=504, p2=34, p3=0, p4=-1

Breakpoint 3, 0x00007fc20bb9d030 in **semop** () from /lib64/libc.so.6
semid=163840
{sem_num=17, sem_op= -1, sem_flg= 0}
nsops=1

kslwt_update_stats_int

```
(gdb) info registers
……..
rbp            0x7fff111e6420 0x7fff111e6420
rsp            0x7fff111e63d0 0x7fff111e63d0
r8            0x7acc7b20      2060221216
r9            0x7acc6bc8      2060217288
r10            0x7f77fe944280 140153348964992
r11            0x7b0f5028      2064601128
r12            0x0        0
r13            0x1f       31
r14            0x7be987f0      2078902256
r15            0x7acc7b20      2060221216
```

```
SELECT to_char(value,'xxxxxxx') ,NAME
 FROM v$sesstat ses_stat
    , v$statname s
 WHERE ses_stat.statistic#= s.statistic#
   AND NAME IN ('redo writes','redo block
   AND VALUE!=0
```

	TO_CHAR(VALUE,'XXXXXXX')	NAME
1	867	redo writes

```
(gdb) x/w 0x7be987f0
0x7be987f0:    0x00000867
(gdb)
```

kcrfw_gather_lwn

(gdb) c
Continuing.
Hardware access (read/write) watchpoint 306: *0x60027c38

Value = 17225
0x0000000002d3fa26 in kcrfw_gather_lwn ()

```
SELECT *  from X$KCRFWS;
```

	ADDR	INDX	INST_ID	CON_ID	NEXT_BLK	LAST_BLK
1	0000000060027C38	0	1	0	17225	17224

Tools

1. strace - trace system calls
2. gdb - GNU debugger
3. systemTap (stap) is a scripting language and tool for dynamically instrumenting running production Linux kernel-based operating systems

strace

```
BEGIN
 dbms_lock.sleep(100);
END;
/

……..
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426775, 171037}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426778, 176509}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426781, 177203}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426784, 178806}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426787, 178494}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426790, 182340}, NULL) = 0
semtimedop(720898, 0xbf842588, 1, {3, 0}) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1440426793, 186095}, NULL) = 0
…….
```

Gdb could be used to stop/slow down LGWR

```
(gdb) b io_submit
Breakpoint 1 at 0xaef560
(gdb) commands
Type commands for when breakpoint 1 is hit, one per line.
End with a line saying just "end".
>shell sleep 10
>c
>end
(gdb) c
Continuing.

Breakpoint 1, 0x00aef560 in io_submit () from /usr/lib/libaio.so.1

Breakpoint 1, 0x00aef560 in io_submit () from /usr/lib/libaio.so.1

Breakpoint 1, 0x00aef560 in io_submit () from /usr/lib/libaio.so.1
…..
…..
```

Systemtap script

```
……..
probe syscall.semctl {
  if (pid() == target()) {
     printf ("%s %s\n",name,argstr)
  }
}
probe syscall.semop,syscall.semtimedop {
  if (pid() == target()) {
    sembuf_sz = 6;
    printf("%s semid= %d\n",name, semid)
    for(i = 0; i < nsops; i++) {
      offset = i * sembuf_sz;
      pointer = sops_uaddr + offset;
      num_addr = & @cast(pointer, "struct sembuf")->sem_num;
      op_addr  = & @cast(pointer, "struct sembuf")->sem_op;
      flg_addr  = & @cast(pointer, "struct sembuf")->sem_flg;
      num = user_short(num_addr);
      sem_op = user_short(op_addr);
      sem_flg = user_short(flg_addr);
      tv_sec_addr = & @cast(timeout_uaddr, "struct timespec")->tv_sec;
      tv_sec  = user_short(tv_sec_addr);
      tv_nsec_addr = & @cast(timeout_uaddr, "struct timespec")->tv_nsec;
      tv_nsec  = user_short(tv_nsec_addr);
      printf("sem_num = %d sem_op=%d sem_flg =%d tv_sec=%d tv_nsec=%d\n", num,sem_op,sem_flg,tv_sec,tv_nsec);
    }
   printf("%s\n", res);
  }
}
```

Environments

Oracle VirtualBox 4.3.12

Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production

Red Hat Enterprise Linux Server release 7.0

Initial setup

```
“_use_single_log_writer”='TRUE‘
"_in_memory_undo"=FALSE
 filesystemio_options=SETALL
 disk_asynch_io=TRUE


CREATE TABLE LOG_BUFFER
(
  id  NUMBER
, val VARCHAR2(100)
)
```

LGWR (idle) sycalls

```
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
semtimedop semid= 163840
  sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
```

LGWR (idle) + trace

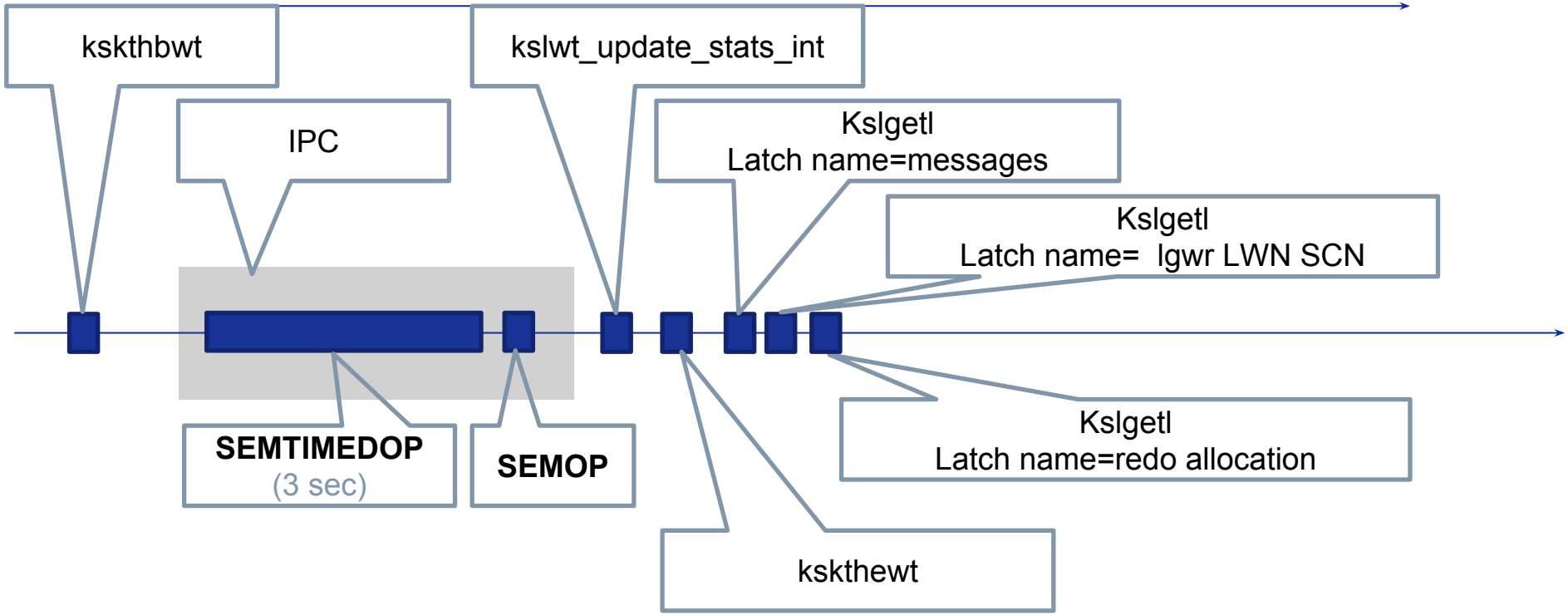
```
semtimedop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
WAIT #0: nam='rdbms ipc message' ela= 3000145 timeout=300 p2=0 p3=0 obj#=-1 tim=3945054915

semtimedop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
WAIT #0: nam='rdbms ipc message' ela= 3001164 timeout=300 p2=0 p3=0 obj#=-1 tim=3948057289

semtimedop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
WAIT #0: nam='rdbms ipc message' ela= 3000353 timeout=300 p2=0 p3=0 obj#=-1 tim=3951058332
```

LGWR(idle) syscalls+oracle functions

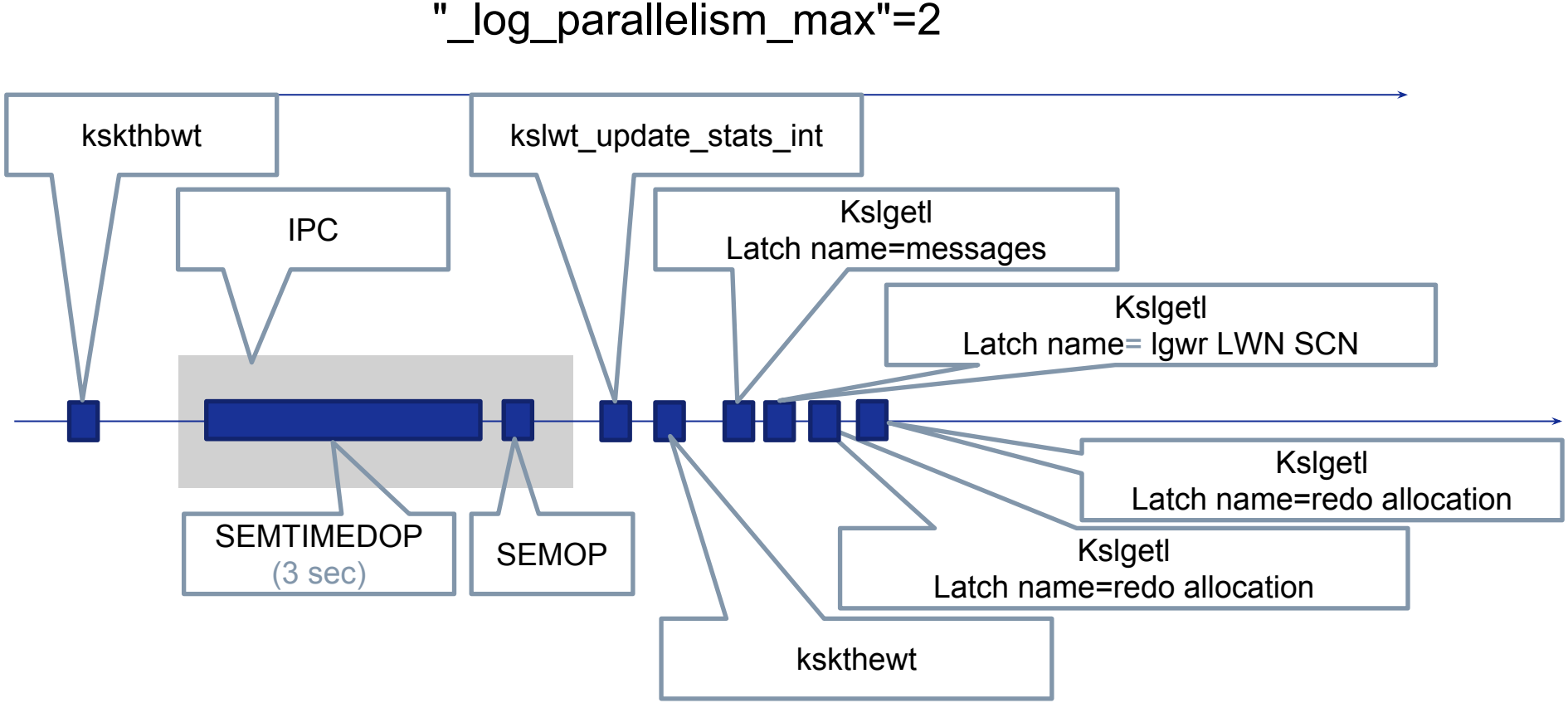
```
kskthbwt event#=8 ,name=rdbms ipc messaget
semtimedop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=3 tv_nsec=0
semop semid= 163840
   sem_num = 17 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
kslwt_update_stats_int rdi=2060221216, rsi=2060221216, rdx=2, rcx=3777893186295716171 r8=1448,
r9=3
kslwt_update_stats_int rdi=2060221216, rsi=2060221216, rdx=30, rcx=237635540964 r8=2060221216,
r9=2055544656
kskthewt event#=8, name=rdbms ipc messaget
kslgetl addr=6000e660, wait=1, where=0, why=223
latch name = messages
kslgetl addr=6001f750, wait=1, where=0, why=1873
latch name = lgwr LWN SCN
ksl_get_shared_latch addr=60031848, wait=0, where=0, why=3952
latch name = KTF sga latch
kslgetl addr=77126af0, wait=1, where=0, why=2781
latch name = redo allocation
kslgetl addr=6000e660, wait=1, where=2065657856, why=222
latch name = messages
```

LGWR (idle) diagram

"_log_parallelism_max"=1



LGWR (idle) diagram



LGWR + 324 bytes of redo

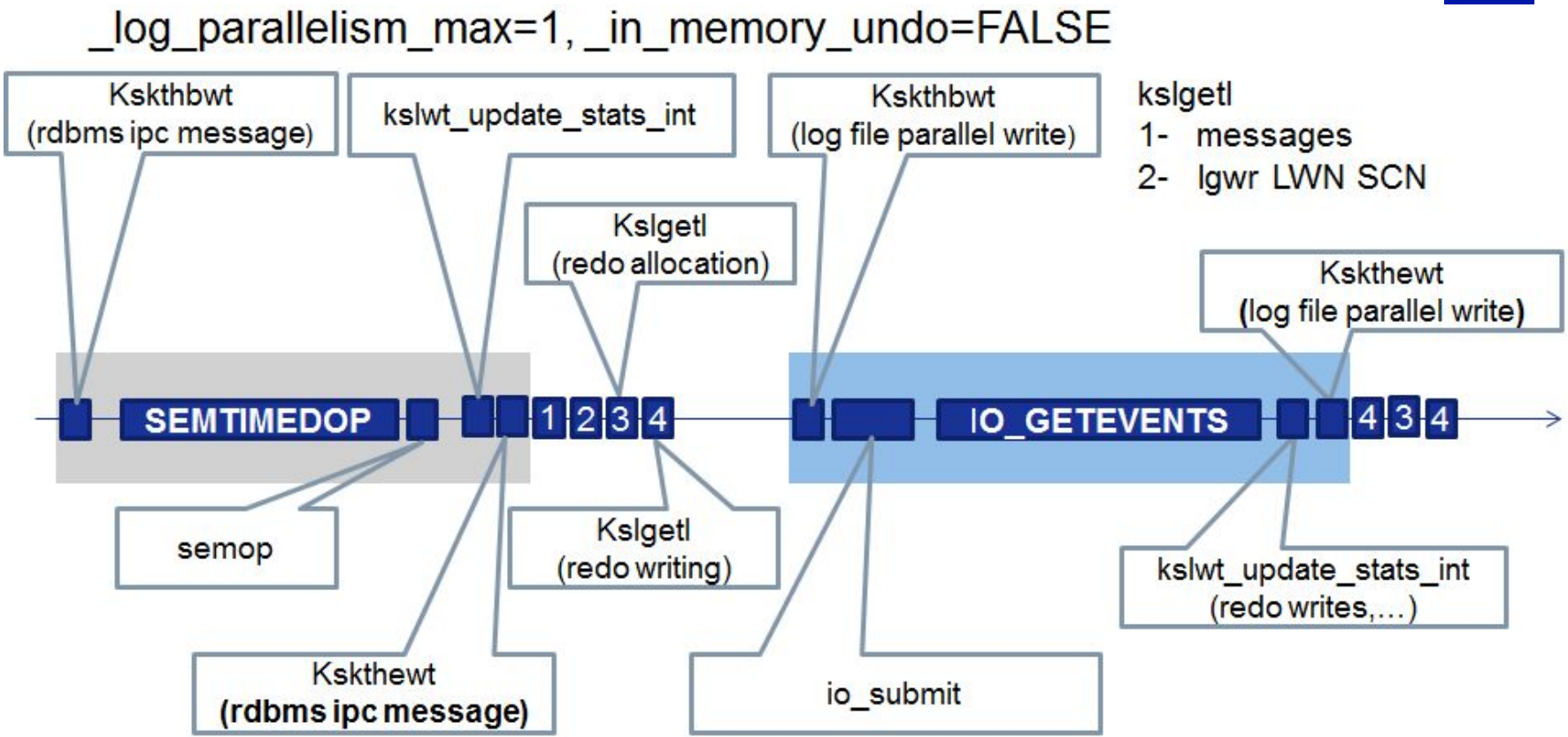
_in_memory_undo=FALSE

```
1 row created.


Execution Plan
----------------------------------------------------------

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT         |            |     1 |   100 |     1   (0)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL | LOG_BUFFER |       |       |            |          |
-------------------------------------------------------------------------------------------


Statistics
----------------------------------------------------------
         0  recursive calls
         3  db block gets
         1  consistent gets
         0  physical reads
       324  redo size
       560  bytes sent via SQL*Net to client
       741  bytes received via SQL*Net from client
         3  SQL*Net roundtrips to/from client
         1  sorts (memory)
         0  sorts (disk)
         1  rows processed
```

LGWR + 324 bytes of redo
_in_memory_undo=FALSE

kskthewt event#=8, name=rdbms ipc message
kslgetl name = messages addr=6000e660, wait=1, where=0, why=223
kslgetl name = lgwr LWN SCN addr=6001f750, wait=1, where=0, why=1873
ksl_get_shared_latch addr=60031848, wait=0, where=0, why=3952
latch name = KTF sga latch
kcrfw_gather_lwn rdi=140733801975296, rsi=2353157160, rdx=0, rcx=140174734742144 r8=0, r9=0
kslgetl name = redo allocation addr=8cd26af0, wait=1, where=0, why=2781
kslgetl name = redo writing addr=60027718, wait=1, where=0, why=2796
kskthbwt event#=137 ,name=log file parallel write
kernel.function("**sys_io_submit**@fs/aio.c:1837").call(+)ora_lgwr_cdb1 ctx_id=0x7f7cf946d000 nr=1
block_id =1723 blocks=1 file=redo01.log op_code=1 offset= 882176 **bytes=512**
kernel.function("**sys_io_submit**@fs/aio.c:1837").return(-) retstr=1
kernel.function("sys_io_getevents@fs/aio.c:1935").call?(+) ctx_id =0x7f7cf946d000 min_nr=1 nr=128 tv_sec=600 tv_nsec=0
kernel.function("sys_io_getevents@fs/aio.c:1935").return?(-) $return=1
kslwt_update_stats_int
kskthewt event#=137, name=log file parallel write
kslgetl name = redo writing addr=60027718, wait=1, where=1723, why=2797
kslgetl name = redo allocation addr=8cd26af0, wait=1, where=0, why=2783
kslgetl name = redo writing addr=60027718, wait=1, where=0, why=2795
……

LGWR +some redo diagram



LOG FILE parallel write parameters (Doc ID 34583.1)

P1 = files
P2 = blocks
P3 = requests

Files - Number of files written to.
If you have more than one log member per group then the files are written to in parallel (if possible). This is the number of redo log members (files) that the writes apply to.

blocks -Number of blocks to be written.
The number of REDO blocks being written to each log member. eg: A value of 10 indicates 10 redo blocks for each log member.

requests - Number of I/O requests.
The number of distinct IO requests.

```
select event, p1,p2,p3 from v$session where sid=81
```

ery Result

SQL | All Rows Fetched: 1 in 0.004 seconds

	EVENT	P1	P2	P3
1	log file parallel write	2	12210	6

kernel.function("sys_io_submit@fs/aio.c:1837").call(+)ora_lgwr_cdb1 ctx_id=0x7f8b2336d000 **nr=6**
addr=0x91c04a00 addr2=2445298176
fd=260 file=**redo03_1.log** op_code=1 offset= 19968 bytes=**1048576**
addr=0x91c04a00 addr2=2445298176
fd=261 file=**redo03_2.log** op_code=1 offset= 19968 bytes=1048576
addr=0x91d04a00 addr2=2446346752
fd=260 file=redo03_1.log op_code=1 offset= 1068544 bytes=**1048576**
addr=0x91d04a00 addr2=2446346752
fd=261 file=redo03_2.log op_code=1 offset= 1068544 bytes=1048576
addr=0x91e04a00 addr2=2447395328
fd=260 file=redo03_1.log op_code=1 offset= 2117120 bytes=**1028608**
addr=0x91e04a00 addr2=2447395328
fd=261 file=redo03_2.log op_code=1 offset= 2117120 bytes=1028608

2*(1048576+1048576+1028608)/512 = 12210

Warning: log write elapsed time

```
Warning: log write elapsed time 10022ms, size 2KB

*** 2016-01-26 04:49:13.798
Warning: log write elapsed time 10018ms, size 5KB

*** 2016-01-26 04:49:38.834
Warning: log write elapsed time 10027ms, size 2KB

*** 2016-01-26 04:49:48.867
Warning: log write elapsed time 10032ms, size 0KB

*** 2016-01-26 04:49:58.906
Warning: log write elapsed time 10037ms, size 10KB

*** 2016-01-26 04:50:08.953
Warning: log write elapsed time 10044ms, size 4KB
```

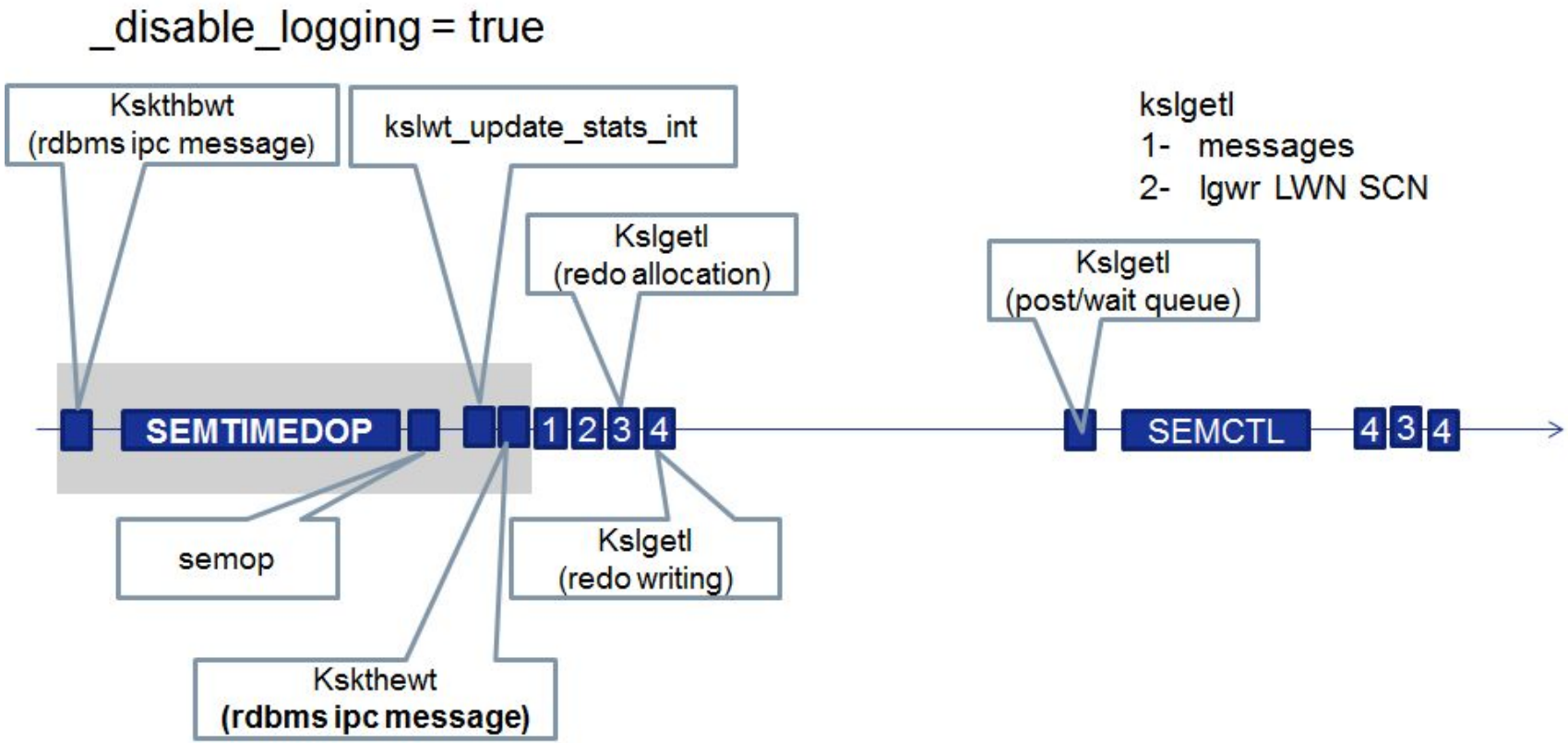
```
(gdb) b io_submit
Breakpoint 1 at 0x7f6511c1b690
(gdb) command
Type commands for breakpoint(s) 1, one per line.
End with a line saying just "end".
>shell sleep 10
>c
>end
(gdb) c
```

_disable_logging=TRUE

Warning, this could be dangerous!

```
kskthewt event#=8, name=rdbms ipc message
kslgetl name = messages addr=6000e660, wait=1, where=0, why=223
kslgetl name = lgwr LWN SCN addr=6001f750, wait=1, where=0, why=1873
ksl_get_shared_latch name=KTF sga latch addr=60031848, wait=0, where=0, why=3952
kcrfw_gather_lwn rdi=140736703090848, rsi=2353157160, rdx=0, rcx=140558936289920 r8=0, r9=0
kslgetl name = redo allocation addr=8cd26af0, wait=1, where=0, why=2781
kslgetl name = redo writing addr=60027718, wait=1, where=0, why=2796
kcscur3 rdi=1610775704, rsi=140736703090236, rdx=1, rcx=0 r8=158, r9=1610774112
kslgetl name = redo writing addr=60027718, wait=1, where=2050, why=2797
kslgetl name = redo allocation addr=8cd26af0, wait=1, where=0, why=2783
kslgetl name = redo writing addr=60027718, wait=1, where=0, why=2795
kslgetl name = Consistent RBA addr=6001fb10, wait=1, where=0, why=1878
kslgetl name = log write info addr=60027af0, wait=0, where=0, why=2822
kslgetl name = post/wait queue addr=912df180, wait=1, where=248, why=14
kslgetl name = post/wait queue addr=912df240, wait=1, where=249, why=15
semctl 163840, 62, SETVAL
kslgetl name = messages addr=6000e660, wait=1, where=2430562304, why=222
kskthbwt event#=8 ,name=rdbms ipc message
```

LGWR diagram

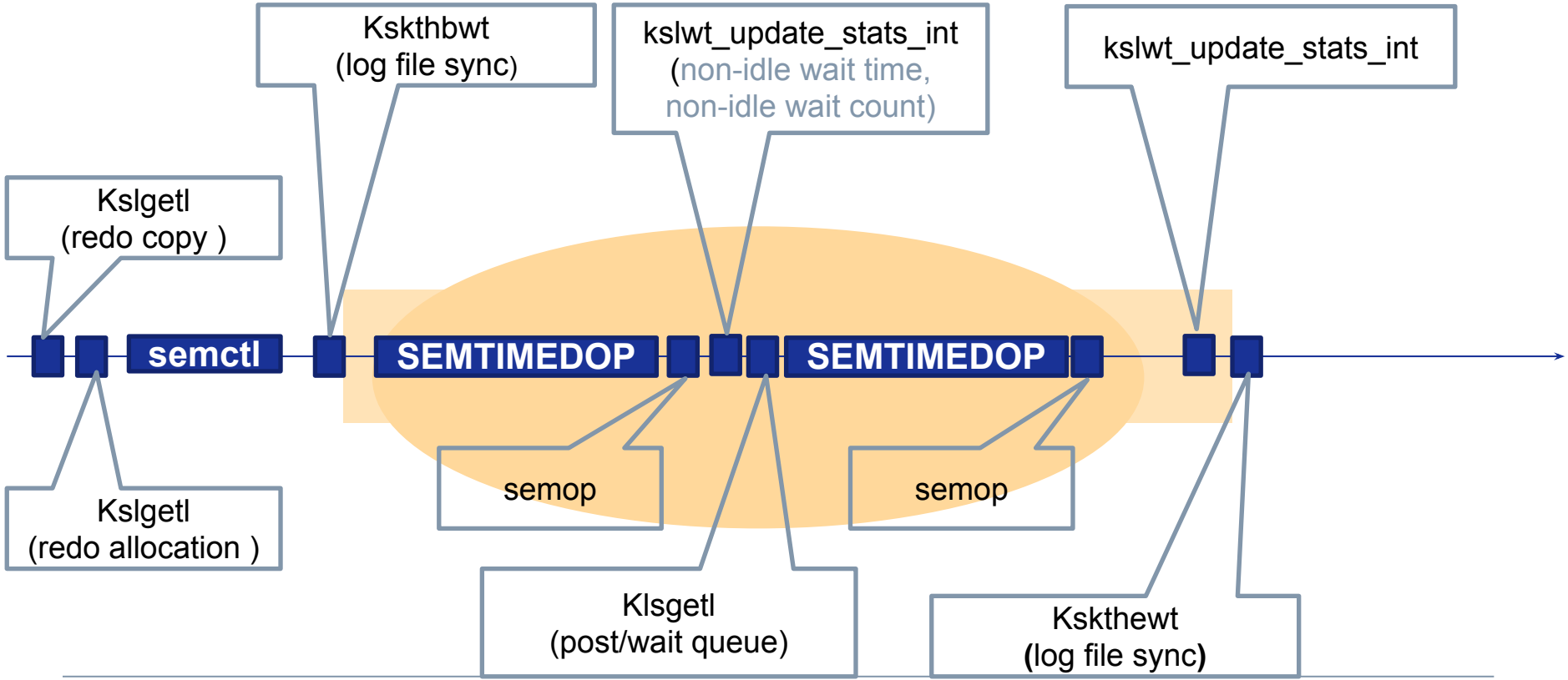


User process+ COMMIT
_use_adaptive_log_file_sync =FALSE

kslgetl name = redo copy addr=8cd22948, wait=0, where=0, why=2808
kslgetl name = redo allocation addr=8cd26af0, wait=0, where=0, why=2776
kslgetl name = messages addr=6000e660, wait=1, where=0, why=227
semctl 163840, 17, SETVAL
kslgetl name = enqueue hash chains addr=903e23c8, wait=1, where=2419784584, why=50
kslgetl name = undo global data addr=8ccada28, wait=1, where=3, why=3779
kslgetl name = enqueue hash chains addr=903e2328, wait=1, where=2419758400, why=50
kslgetl name = DML lock allocation addr=8d257c68, wait=1, where=2367322920, why=3724
kslgetl name = post/wait queue addr=912df180, wait=0, where=248, why=16
kskthbwt event#=146 ,name=log file sync
semtimedop semid= **163840**
  sem_num = **86** sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=-7936
semop semid= 163840
  sem_num = **86** sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
kslwt_update_stats_int
kslgetl name = shared pool addr=6010fcf0, wait=1, where=2022404944, why=4678
…
kslwt_update_stats_int
kskthewt event#=146, name=log file sync

User process diagram

_use_adaptive_log_file_sync =FALSE



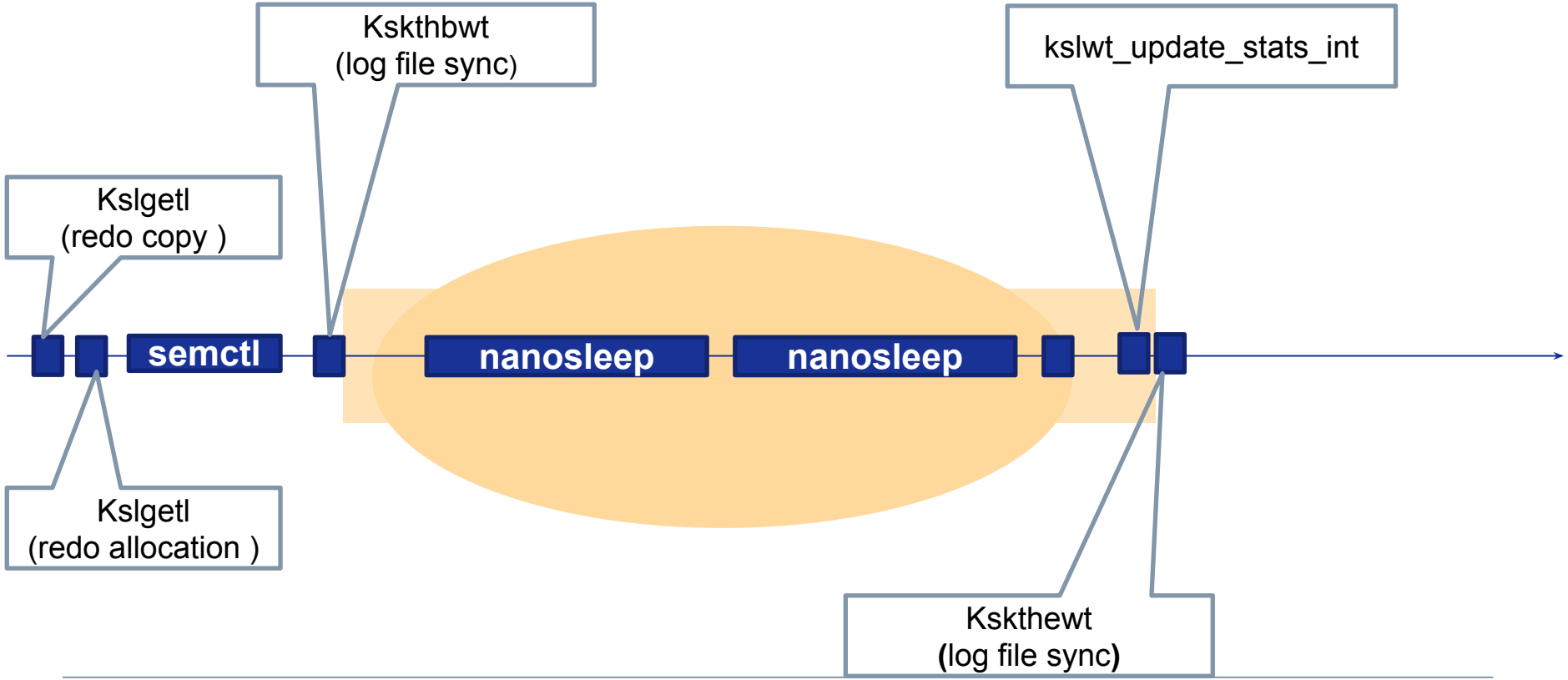
User process+ COMMIT

_use_adaptive_log_file_sync ='POLLING_ONLY'

kslgetl name = redo copy addr=8cd22948, wait=0, where=0, why=2808
kslgetl name = redo allocation addr=8cd26af0, wait=0, where=0, why=2776
kslgetl name = messages addr=6000e660, wait=1, where=0, why=227
semctl 163840, 17, SETVAL
kslgetl name = enqueue hash chains addr=903e23c8, wait=1, where=2419803784, why=50
kslgetl name = undo global data addr=8ccadac8, wait=1, where=4, why=3779
kslgetl name = enqueue hash chains addr=903e2328, wait=1, where=2419758400, why=50
kslgetl name = DML lock allocation addr=8d257c68, wait=1, where=2367322920, why=3724
kskthbwt event#=146 ,name=log file sync
nanosleep sec=0 nanosec=26952
nanosleep sec=0 nanosec=26952
nanosleep sec=0 nanosec=26952
.....

User process diagram

_use_adaptive_log_file_sync ='POLLING_ONLY'



Log file sync switching

*** 2016-02-03 20:26:52.520
Log file sync switching to polling
Current scheduling delay is 1 usec
Current approximate redo synch write rate is 1 per sec
kcrfw_update_adaptive_sync_mode: poll->post current_sched_delay=0 switch_sched_delay=1
current_sync_count_delta=1 switch_sync_count_delta=5

*** 2016-02-03 21:38:24.449
Log file sync switching to post/wait
Current approximate redo synch write rate is 0 per sec

*** 2016-02-03 21:58:26.212
Warning: log write elapsed time 546ms, size 30KB

*** 2016-02-03 22:00:59.902
Warning: log write elapsed time 557ms, size 94KB

*** 2016-02-03 22:01:12.972

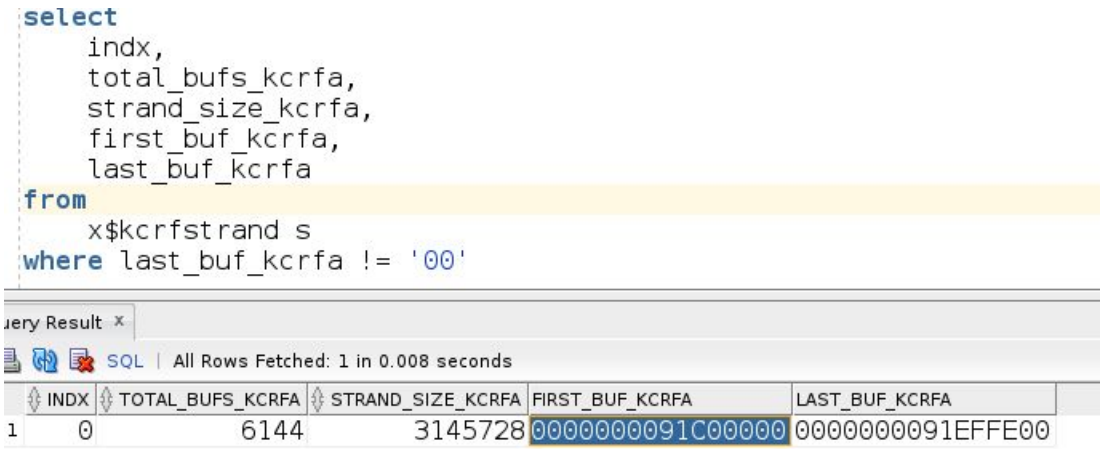
LOG FILE SYNC PARAMETERS

P1 = buffer#
P2 = Not used / sync scn
P3 = Not used
buffer#
All changes up to this buffer number (in the log buffer) must be flushed to disk and the writes confirmed to ensure that the transaction is committed , and will remain committed upon an instance crash. Hence the wait is for LGWR to flush up to this buffer#.
sync scn (11.2 onwards)
Base of the SCN value required to be synced to disk. Hence the wait is for LGWR to flush up to this SCN base.

oradebug ipc

```
Area #0 `Fixed Size' containing Subareas 2-2
 Total size 00000000002cb510 Minimum Subarea size 00000000
  Area  Subarea    Shmid    Segment Addr    Stable Addr    Actual Addr
    0        2   655360 0x00000060000000 0x00000060000000 0x00000060000000
           Subarea size     Segment size   Req_Protect  Cur_protect
                    00000000002cc000 00000000002cc000 default       readwrite
Area #1 `Variable Size' containing Subareas 0-0
 Total size 0000000031800000 Minimum Subarea size 00400000
  Area  Subarea    Shmid    Segment Addr    Stable Addr    Actual Addr
    1        0   688129 0x00000060400000 0x00000060400000 0x00000060400000
           Subarea size     Segment size   Req_Protect  Cur_protect
                    0000000031800000 0000000031800000 default       readwrite
Area #2 `Redo Buffers' containing Subareas 1-1
 Total size 0000000000534000 Minimum Subarea size 00001000
  Area  Subarea    Shmid    Segment Addr    Stable Addr    Actual Addr
    2        1   720898 0x00000091c00000 0x00000091c00000 0x00000091c00000
           Subarea size     Segment size   Req_Protect  Cur_protect
                    0000000000534000 0000000000534000 default       readwrite
```

X$KCRFSTRAND

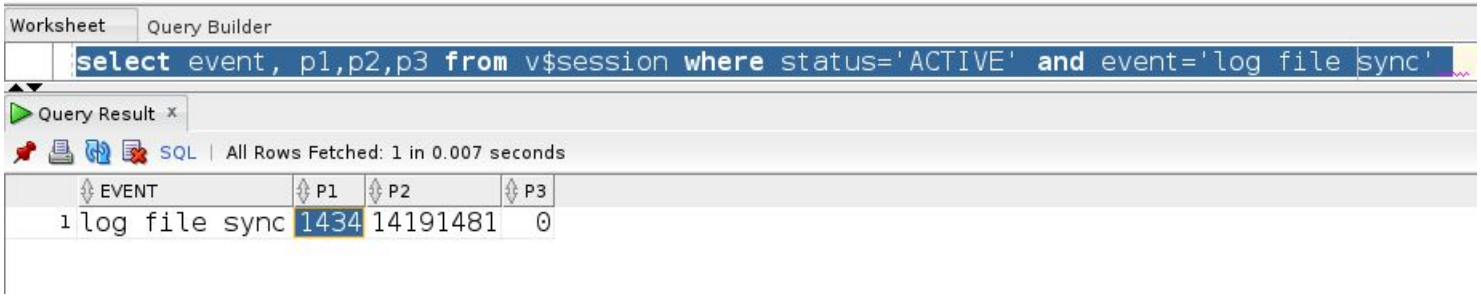


```
select
    indx,
    total_bufs_kcrfa,
    strand_size_kcrfa,
    first_buf_kcrfa,
    last_buf_kcrfa
from
    x$kcrfstrand s
where last_buf_kcrfa != '00'
```

uery Result

SQL | All Rows Fetched: 1 in 0.008 seconds

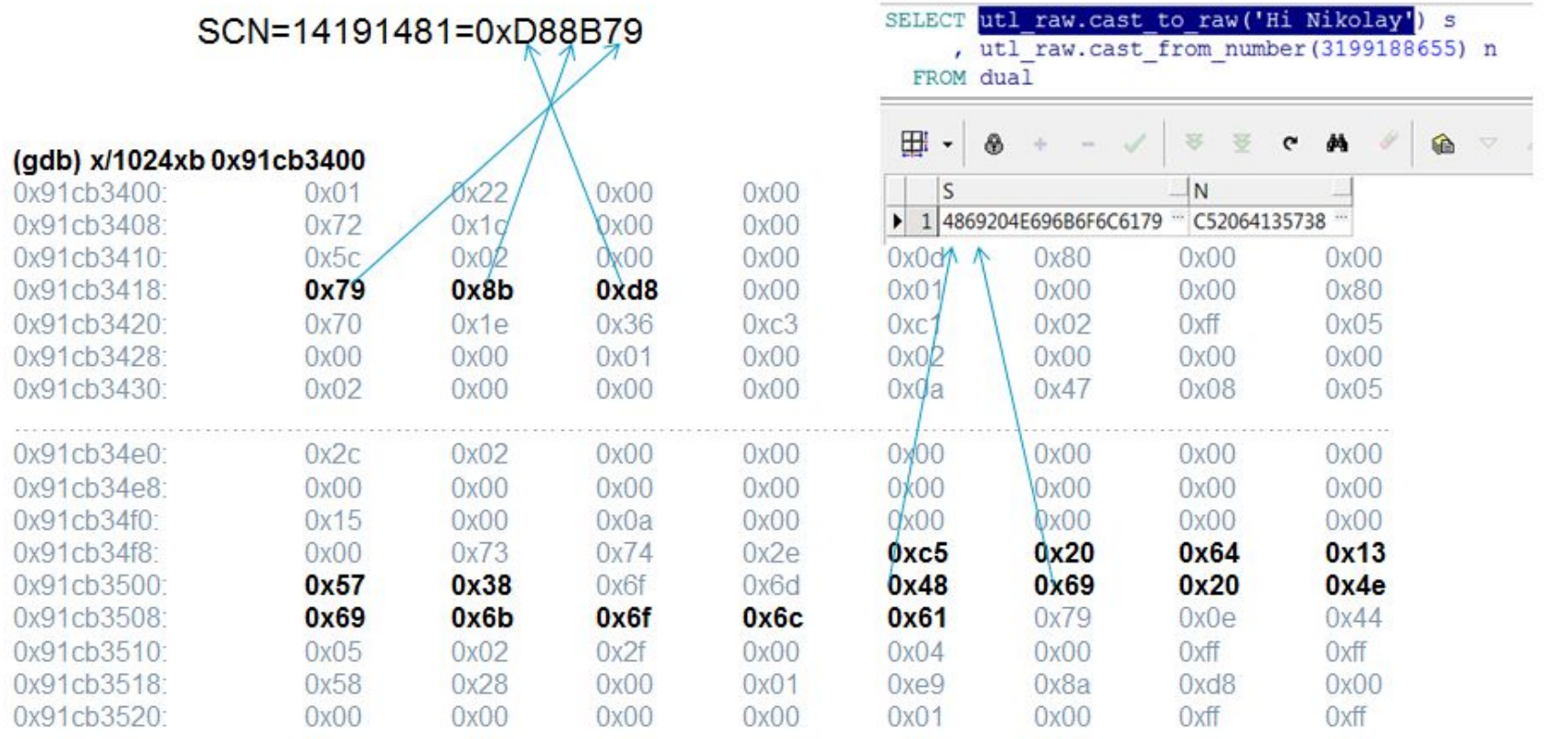
	INDX	TOTAL_BUFS_KCRFA	STRAND_SIZE_KCRFA	FIRST_BUF_KCRFA	LAST_BUF_KCRFA
1	0	6144	3145728	0000000091C00000	0000000091EFFE00



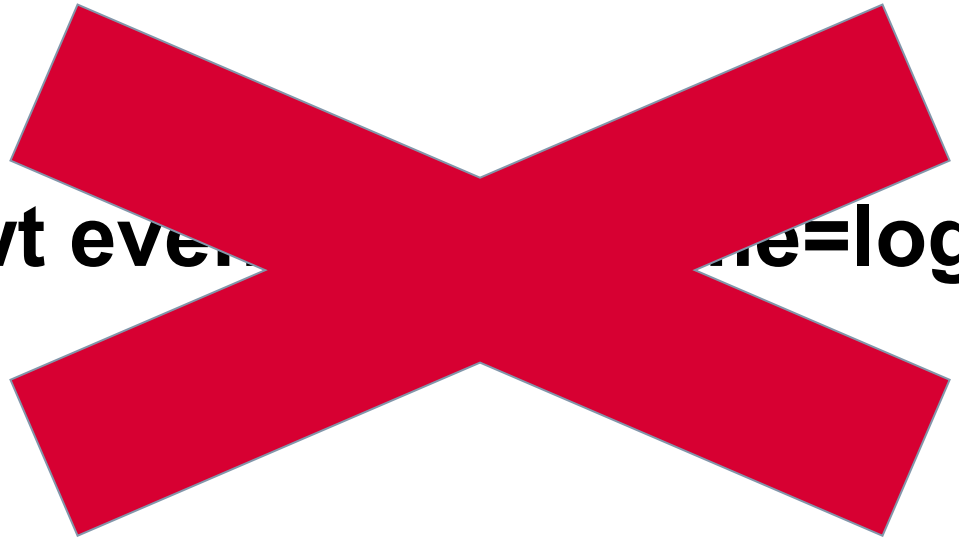
kernel.function("sys_io_submit@fs/aio.c:1837").call(+)ora_lgwr_cdb1 ctx_id=0x7fd8ba183000 nr=1
addr=**0x91cb3400** addr2=2446013440
fd=258 file=redo01.log op_code=1 offset= 8151040 bytes=1024

to_char(to_number('91C00000','XXXXXXXX') + (1434)*512,'XXXXXXXX') = **91CB3400**

LOG BUFFER memory DUMP



NO “log file sync” in some cases on commit

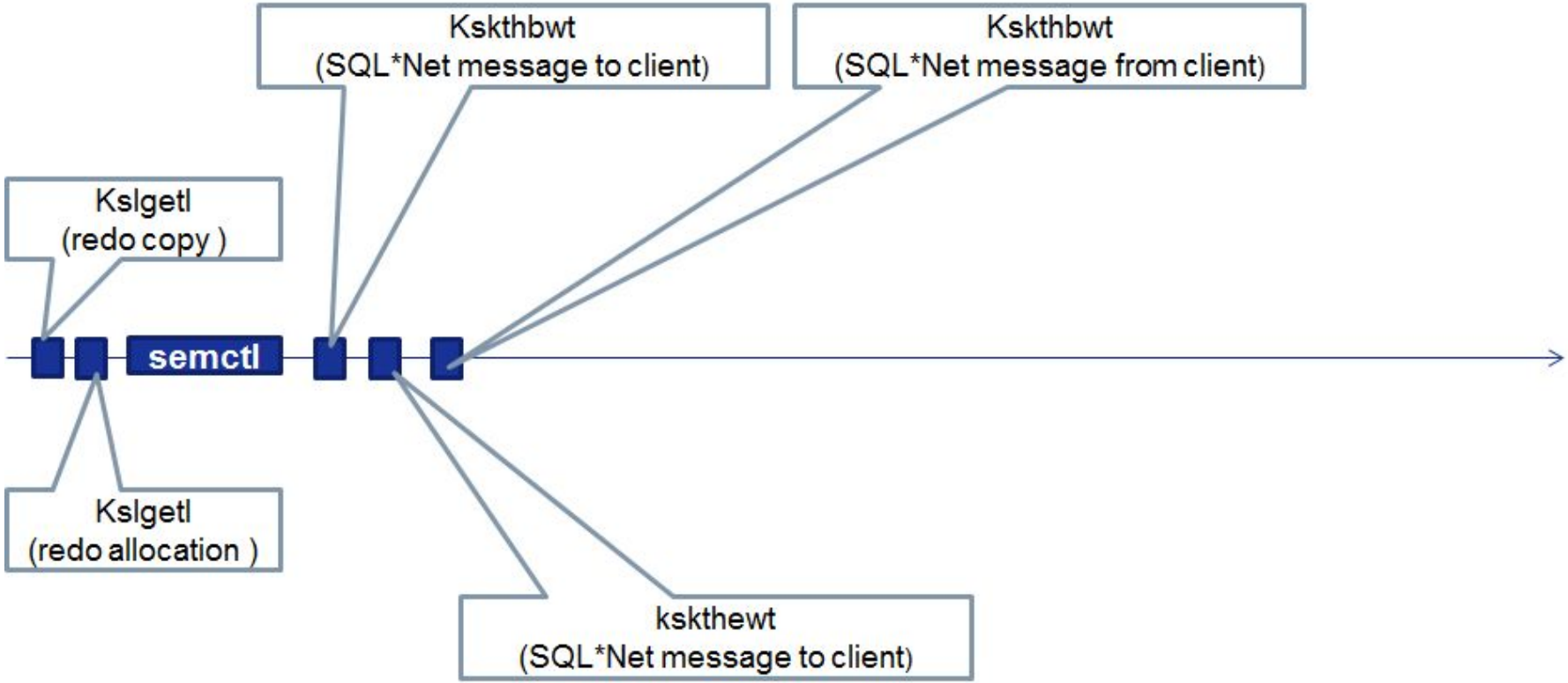


kskthbwt eve[illegible]e=log file sync

commit write nowait immediate

***pread block_id =0 blocks=16 file=TCP
kslwt_update_stats_int
kskthewt event#=388, name=SQL*Net message from client
ksl_get_shared_latch addr=904edde8, wait=1, where=2427087336, why=115 latch name = session idle bit
ksl_get_shared_latch addr=904edde8, wait=1, where=2427087336, why=116 latch name = session idle bit
ksl_get_shared_latch addr=904edde8, wait=1, where=2427087336, why=115 latch name = session idle bit
kslgetl name = session allocation addr=90f20fe8, wait=0, where=4, why=101
kslgetl name = **redo copy** addr=8cd22948, wait=0, where=0, why=2808
kslgetl name = **redo allocation** addr=8cd26af0, wait=0, where=0, why=2776
kslgetl name = messages addr=6000e660, wait=1, where=0, why=227
semctl 163840, 17, SETVAL
kslgetl name = enqueue hash chains addr=903e2468, wait=1, where=2419818040, why=50
kslgetl name = undo global data addr=8ccad988, wait=1, where=9, why=3779
kslgetl name = enqueue hash chains addr=903e2328, wait=1, where=2419793008, why=50
kslgetl name = DML lock allocation addr=8d258000, wait=1, where=2367681680, why=3724
ksl_get_shared_latch addr=904edde8, wait=1, where=2427036936, why=111 latch name = session idle bit
kslgetl name = session allocation addr=90f20fe8, wait=1, where=0, why=105
ksl_get_shared_latch addr=904edde8, wait=1, where=2427087336, why=116 latch name = session idle bit
kskthbwt event#=384 ,name=SQL*Net message to client
kslwt_update_stats_int
kskthewt event#=384, name=SQL*Net message to client
kskthbwt event#=388 ,name=SQL*Net message from client
kernel.function("sys_read@fs/read_write.c:499").call
***pread block_id =0 blocks=16 file=TCP

User process diagram
commit write nowait



LOOP and COMMIT

```
kslgetl name = redo copy addr=8cd22870, wait=0, where=0, why=2808
kslgetl name = redo allocation addr=8cd26af0, wait=0, where=0, why=2776
……..
semctl 163840, 17, SETVAL
kslgetl name = enqueue hash chains addr=903e2328, wait=1, where=2419742104,
why=50
kslgetl name = undo global data addr=8ccad848, wait=1, where=7, why=3779
kslgetl name = enqueue hash chains addr=903e25a8, wait=1, where=2419670512,
why=50
kslgetl name = DML lock allocation addr=8d258170, wait=1, where=2367825184,
why=3724
ksl_get_shared_latch name=session idle bit addr=904edc08, wait=1,
where=2426390136, why=111
kslgetl name = session allocation addr=90f20f40, wait=1, where=0, why=105
kskthbwt event#=402 ,name=PL/SQL lock timer
semtimedop semid= 163840
   sem_num = 85 sem_op=-1 sem_flg =0 tv_sec=1 tv_nsec=0
semop semid= 163840
   sem_num = 85 sem_op=-1 sem_flg =0 tv_sec=0 tv_nsec=0
kslwt_update_stats_int
kslwt_update_stats_int
kskthewt event#=402, name=PL/SQL lock timer
```

```
BEGIN
  FOR I_i IN 1..3
  LOOP
    UPDATE log_buffer
      SET val=val;
    COMMIT;
    dbms_lock.sleep(1);
  END LOOP;
END;
```

When LGWR is fast enough

semctl 163840, 17, SETVAL

kslgetl name = enqueue hash chains addr=903e2508, wait=1, where=2419725064, why=50

kslgetl name = undo global data addr=8ccad988, wait=1, where=9, why=3779

kslgetl name = enqueue hash chains addr=903e25a8, wait=1, where=2419670512, why=50

kslgetl name = DML lock allocation addr=8d258170, wait=1, where=2367825184, why=3724

kcscur3 rdi=1610775704, rsi=140737190756144, rdx=1, rcx=1588 r8=0, r9=0

ksl_get_shared_latch name=session idle bit addr=904edc08, wait=1, where=2426390136, why=111

kslgetl name = session allocation addr=90f20f40, wait=1, where=0, why=105

ksl_get_shared_latch name=session idle bit addr=904ede88, wait=1, where=2426406936, why=116

kskthbwt event#=384 ,name=SQL*Net message to client

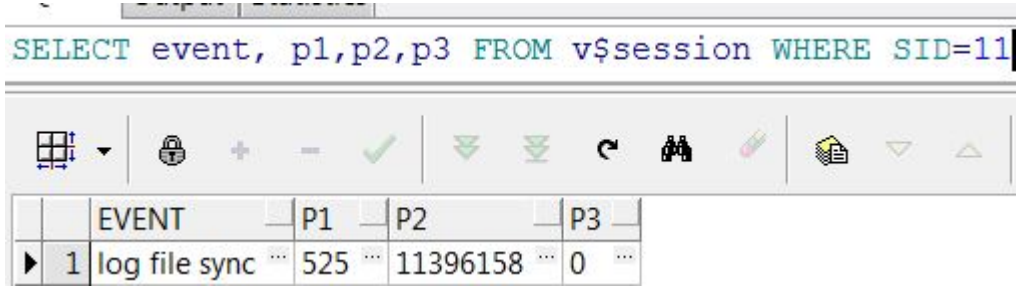
kslwt_update_stats_int

kskthewt event#=384, name=SQL*Net message to client

kskthbwt event#=388 ,name=SQL*Net message from client

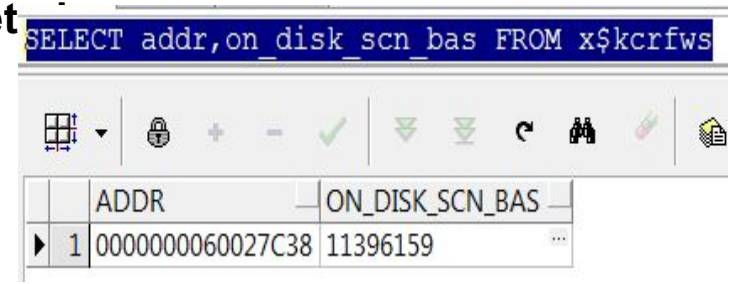
```
(gdb) b semctl
(gdb) command
>shell sleep 10
>c
>end
(gdb) c
Continuing.
```

ON_DISK_SCN_BAS



	EVENT	P1	P2	P3
1	log file sync	525	11396158	0

(gdb) set {int}0x60027C98 = 11396158 – still waiting
(gdb) set {int}0x60027C98 = 11396159 - !!!! complet



	ADDR	ON_DISK_SCN_BAS
1	0000000060027C38	11396159

semtimedop semid= 163840
……
kslwt_update_stats_int
kslgetl name = post/wait queue addr=912df180, wait=1, where=248, why=17
kslwt_update_stats_int
kskthewt event#=146, name=log file sync

Some session statistics

user commits

- it’s increased when session issues commit (even if LGWR is suspended)

redo synch writes

- it’s increased after LGWR had written data to disk

My advice

- Estimate your LFPW waits before any changes
- Commit less
- Generate less redo
- Read Doc ID 34583.1

References

http://blog.tanelpoder.com/

https://fritshoogland.wordpress.com

https://andreynikolaev.wordpress.com

https://jonathanlewis.wordpress.com

https://www.google.ch/patents/US5974425

http://savvinov.com

https://dmitryremizov.wordpress.com

https://orainternals.wordpress.com/2012/02/10/what-is-rdbms-ipc-message-wait-event/

http://www.minek.com/files/unix_examples/semab.html

http://www.cs.cf.ac.uk/Dave/C/node26.html

http://nofxsss2007.narod.ru/spo/47.htm

http://eli.thegreenplace.net/2011/09/06/stack-frame-layout-on-x86-64/

This is not an offer to provide any services. This material is for information and illustrative purposes only and is not intended, nor should it be distributed, for advertizing purposes, nor is it intended for publication or broadcast. Any third party analysis does not constitute any endorsement or recommendation. Opinions expressed herein are current opinions as of the date appearing in this material only and are subject to change without notice This information is provided with the understanding that with respect to the material provided herein, that you will make your own independent decision with respect to any course of action in connection herewith and as to whether such course of action is appropriate or proper based on your own judgment, and that you are capable of understanding and assessing the merits of a course of action. “Deutsche Bank TechCentre” LLC shall not have any liability for any damages of any kind whatsoever relating to this material.