
# Strace, Perf & Gdb – three friends for the DBA

Kamil Stawiarski (@ora600pl)

Database Whisperers sp. z o. o. sp. k.
ul. Płocka 5a
01-231 Warszawa
www.ora-600.pl

# DB FILE SCATTERED READ – MBRC AND ASYNC IO ON LINUX

Recently one of my students asked me if there is any correlation between the AU size in ASM diskgroup and parameter db_file_multiblock_read_count. I made some tests but results are very strange.

I have 4 same tables, located in 4 different diskgroups:

```
  1  select t.table_name,  g.name, t.blocks, g.ALLOCATION_UNIT_SIZE
  2  from dba_tables t, dba_data_files d, v$asm_diskgroup g
  3  where t.tablespace_name=d.tablespace_name
  4  and   g.name=substr(d.file_name,2,instr(d.file_name,'/')-2)
  5  and  t.owner='SH'
  6* and  t.table_name like 'SALES%'
SQL> /

TABLE_NAME NAME                       BLOCKS ALLOCATION_UNIT_SIZE
---------- ------------------------------ ---------- --------------------
SALES16M   DATA16M                 93551         16777216
SALES32M   DATA32M                 93551         33554432
SALES1M    DATA1M                  93551          1048576
SALES4M    DATA4M                  93551          4194304
```

All tables are smaller then a hidden parameter "_small_table_threshold" so table access full on all tables will be using db file scattered read.

```
SQL> ;
  1  select p.ksppinm, v.ksppstvl
  2  from x$ksppi p, x$ksppcv v
  3  where p.indx=v.indx
  4* and p.ksppinm='_small_table_threshold'
SQL> /

KSPPINM                   KSPPSTVL
----------------------------------------- ------------------------------
_small_table_threshold           101527
```

Let's start the main test. Parameter db_file_multiblock_read_count is set to the default value

```
[oracle@rico ~]$ sqlplus sh/sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Jul 17 10:30:36 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Wed Jul 01 2015 11:16:33 +02:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options

SQL> sho parameters db_file_multiblock

NAME                     TYPE    VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     128
SQL>
```

I will identify the server process, correlated with my session

```
SQL> ;
  1  select spid
  2  from v$process p, v$session s
  3  where p.addr=s.paddr
  4* and s.username='SH'
SQL> /

SPID
------------------------
10430
```

Now I will use strace on this server process:

```
[root@rico ~]# strace -c -p 10430
Process 10430 attached - interrupt to quit
```

I will gather results for the following queries:

```
SQL> select count(1) from SALES16M;

  COUNT(1)
----------
  19295703

SQL> select count(1) from SALES32M;

  COUNT(1)
----------
  19295703

SQL> select count(1) from SALES1M;

  COUNT(1)
----------
  19295703

SQL> select count(1) from SALES4M;

  COUNT(1)
----------
  19295703
```

Results for SALES16M

```
[root@rico ~]# strace -c -p 10430
Process 10430 attached - interrupt to quit
^CProcess 10430 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
100.00    0.053000          71       750           pread
  0.00    0.000000           0        31           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        16         7 open
  0.00    0.000000           0         9           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0         1           fstat
  0.00    0.000000           0        28           lseek
  0.00    0.000000           0         3           mmap
  0.00    0.000000           0         2           munmap
```

```
  0.00    0.000000           0         1           brk
  0.00    0.000000           0         1           uname
  0.00    0.000000           0        10           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        28           times
  0.00    0.000000           0         1           getuid
  0.00    0.000000           0         1           io_getevents
  0.00    0.000000           0         1           io_submit
------ ----------- ----------- --------- --------- ----------------
100.00    0.053000                  1051         7 total
```

Results for SALES32M

```
[root@rico ~]# strace -c -p 10430
Process 10430 attached - interrupt to quit
^CProcess 10430 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 98.16    0.053462          72       747           pread
  1.84    0.001000         500         2           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         1           open
  0.00    0.000000           0         1           close
  0.00    0.000000           0         1           mmap
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        28           times
------ ----------- ----------- --------- --------- ----------------
100.00    0.054462                   950           total
```

Results for SALES1M

```
[root@rico ~]# strace -c -p 10430
Process 10430 attached - interrupt to quit
^CProcess 10430 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 48.58    0.015000          26       583           io_getevents
 39.99    0.012349          75       164           pread
 11.43    0.003528           6       583           io_submit
  0.00    0.000000           0         3           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         2           open
  0.00    0.000000           0         2           close
  0.00    0.000000           0         1           semctl
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        29           times
  0.00    0.000000           0         3           semtimedop
------ ----------- ----------- --------- --------- ----------------
100.00    0.030877                  1540           total
```

Results for SALES4M

```
[root@rico ~]# strace -c -p 10430
Process 10430 attached - interrupt to quit
```

```
^CProcess 10430 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 95.16    0.043000          65       664           pread
  2.62    0.001186          14        83           io_submit
  2.21    0.001000          12        83           io_getevents
  0.00    0.000000           0         2           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        21           getrusage
  0.00    0.000000           0        12           times
------ ----------- ----------- --------- --------- ----------------
100.00    0.045186                   867           total
```

As we can see, the ASYNC IO represented by syscalls: *io_submit* and *io_geteventsare* mostly used while scanning table located in diskgroup with 1MB AU.

At first I thought that this is correlated with *db_file_multiblock_read_count*. The default value (128) multiplied by 8k (block size for my database) is exactly 1M. So I made a few more tests with altered values for this parameter. First results where promising – I have managed to get ASYNC IO for all of the tables.

But then I made the final (a little stupid :)) test. After creating a session I have altered value of the *db_file_multiblock_read_count* to 128. Yes: 128. So the same as default:

```
[oracle@rico ~]$ sqlplus sh/sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Jul 17 10:49:25 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Fri Jul 17 2015 10:48:43 +02:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options

SQL> sho parameters db_file_multiblock

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_file_multiblock_read_count        integer     128
SQL> alter session set db_file_multiblock_read_count=128;

Session altered.
```

Let's check the results now:

SALES16M

```
[root@rico ~]# strace -c -p 11208
Process 11208 attached - interrupt to quit
^CProcess 11208 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 53.96    0.007032           9       747           io_submit
 46.04    0.006000           8       747           io_getevents
  0.00    0.000000           0        31           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        16         7 open
  0.00    0.000000           0         9           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0         1           fstat
```

```
  0.00    0.000000           0        28           lseek
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         1           brk
  0.00    0.000000           0         4           pread
  0.00    0.000000           0         1           uname
  0.00    0.000000           0        10           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        34           times
  0.00    0.000000           0         1           getuid
------ ----------- ----------- --------- --------- ----------------
100.00    0.013032                  1807         7 total
```

SALES32M

```
[root@rico ~]# strace -c -p 11208
Process 11208 attached - interrupt to quit
^CProcess 11208 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 54.18    0.008278          11       746           io_submit
 45.82    0.007000           9       746           io_getevents
  0.00    0.000000           0         2           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         1           open
  0.00    0.000000           0         1           close
  0.00    0.000000           0         1           pread
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        28           times
------ ----------- ----------- --------- --------- ----------------
100.00    0.015278                  1695           total
```

SALES1M

```
[root@rico ~]# strace -c -p 11208
Process 11208 attached - interrupt to quit
^CProcess 11208 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 65.36    0.010000          13       749           io_getevents
 34.64    0.005300           7       746           io_submit
  0.00    0.000000           0         3           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         2           open
  0.00    0.000000           0         2           close
  0.00    0.000000           0         1           pread
  0.00    0.000000           0         1           semctl
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0        28           times
  0.00    0.000000           0         2           semtimedop
------ ----------- ----------- --------- --------- ----------------
100.00    0.015300                  1704           total
```

SALES4M

```
[root@rico ~]# strace -c -p 11208
Process 11208 attached - interrupt to quit
^CProcess 11208 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 50.10    0.006000           8       746           io_getevents
 49.90    0.005975           8       746           io_submit
  0.00    0.000000           0         2           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0         1           pread
  0.00    0.000000           0        21           getrusage
  0.00    0.000000           0        12           times
------ ----------- ----------- --------- --------- ----------------
100.00    0.011975                  1530           total
```

So EVERYTHING is being read with ASYNC IO!!! And this not the craziest thing – the other sessions are also using ASYNC IO for those tables now. Without altering the parameter. I have tried to flush buffer_cache and shared_pool, but with no success. The situation goes back to square one after restarting the instance.

I occurs that it takes only one session to set the parameter to any value – even the same value as it already was by default. And when the first query will use db file scattered read in async mode – all the other sessions, querying tables on ASM will also use *io_getevents* and *io_submit* instead of *pread*. (!!!)

```
[oracle@rico ~]$ sqlplus sh/sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Jul 17 10:59:22 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Fri Jul 17 2015 10:56:53 +02:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options

SQL> alter session set db_file_multiblock_read_count=128;

Session altered.

SQL> select count(1) from sales16m;

  COUNT(1)
----------
  19295703

SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options
[oracle@rico ~]$ sqlplus sh/sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Jul 17 10:59:51 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Fri Jul 17 2015 10:59:22 +02:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options
```

```
SQL> select count(1) from SALES4M;

  COUNT(1)
----------
  19295703

SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options
[oracle@rico ~]$ sqlplus sh/sh

SQL*Plus: Release 12.1.0.2.0 Production on Fri Jul 17 11:00:43 2015

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Last Successful login time: Fri Jul 17 2015 10:59:51 +02:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options

SQL> select count(1) from sales32m;

  COUNT(1)
----------
  19295703
```

The appropriate *strace* results are:

```
[root@rico ~]# strace -c -p 11782
Process 11782 attached - interrupt to quit
^CProcess 11782 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 53.97    0.006000           8       747           io_getevents
 45.60    0.005069           7       747           io_submit
  0.43    0.000048           1        34           times
  0.00    0.000000           0        31           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        16         7 open
  0.00    0.000000           0         9           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0         1           fstat
  0.00    0.000000           0        28           lseek
  0.00    0.000000           0         7           mmap
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         1           brk
  0.00    0.000000           0         4           pread
  0.00    0.000000           0         1           uname
  0.00    0.000000           0        10           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       164           getrusage
  0.00    0.000000           0         1           getuid
------ ----------- ----------- --------- --------- ----------------
100.00    0.011117                  1806         7 total
[root@rico ~]# strace -c -p 11793
Process 11793 attached - interrupt to quit
^CProcess 11793 detached
% time     seconds  usecs/call     calls    errors syscall
```

```
------ ----------- ----------- --------- --------- ----------------
 51.71    0.007000          10       731           io_getevents
 48.29    0.006536           9       731           io_submit
  0.00    0.000000           0        32           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        17         7 open
  0.00    0.000000           0        10           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0         1           fstat
  0.00    0.000000           0        28           lseek
  0.00    0.000000           0         9           mmap
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         1           brk
  0.00    0.000000           0        17           pread
  0.00    0.000000           0         1           uname
  0.00    0.000000           0         1           semctl
  0.00    0.000000           0        10           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       209           getrusage
  0.00    0.000000           0        50           times
  0.00    0.000000           0         1           getuid
  0.00    0.000000           0         2           semtimedop
------ ----------- ----------- --------- --------- ----------------
100.00    0.013536                1856         7 total
[root@rico ~]# strace -c -p 11823
Process 11823 attached - interrupt to quit
^CProcess 11823 detached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 51.65    0.005342           7       731           io_submit
 48.35    0.005000           7       731           io_getevents
  0.00    0.000000           0        31           read
  0.00    0.000000           0         2           write
  0.00    0.000000           0        16         7 open
  0.00    0.000000           0         9           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0         1           fstat
  0.00    0.000000           0        28           lseek
  0.00    0.000000           0         9           mmap
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         1           brk
  0.00    0.000000           0        17           pread
  0.00    0.000000           0         1           uname
  0.00    0.000000           0        10           fcntl
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0       209           getrusage
  0.00    0.000000           0        50           times
  0.00    0.000000           0         1           getuid
------ ----------- ----------- --------- --------- ----------------
100.00    0.010342                1850         7 total
```

Results from *perf* seams to confirm this strange behaviour.

Before setting the parameter:

```
[root@rico ~]# perf record -p 13521
```

```
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.061 MB perf.data (~2644 samples) ]

[root@rico ~]# perf report | grep aio
     0.06%  oracle_13521_or  [kernel.kallsyms]   [k] do_aio_read
[root@rico ~]# perf report | grep pread
     0.06%  oracle_13521_or  libpthread-2.12.so  [.] __pread_nocancel
```

After setting the parameter:

```
[root@rico ~]# perf record -p 13573
^C[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.061 MB perf.data (~2680 samples) ]

[root@rico ~]# perf report | grep pread
[root@rico ~]# perf report | grep aio
     0.18%  oracle_13573_or  libaio.so.1.0.1     [.] 0x0000000000000615
     0.06%  oracle_13573_or  oracle              [.] ksfdaio
     0.06%  oracle_13573_or  [kernel.kallsyms]   [k] aio_put_req
     0.06%  oracle_13573_or  [kernel.kallsyms]   [k] aio_run_iocb
```

I made tests on: Oracle 12.1.0.2 EE (OEL 6.6) and Oracle 11.2.0.3 EE (OEL 6.3)

So the conclusion is: WTF?!

## ORACLE 12C: PRAGMA UDF – THE TRUTH

Let's check, why PRAGMA UDF makes execution faster, then regular function.
C function responsible for calling a PL/SQL code from SQL is called "plsql_run":

```
[oracle@rico bin]$ nm oracle | grep plsql_run
000000000ce1c090 T plsql_run
```

Now let's try to create a new HR session and create a simple function which will multiple values by 2:

```
SQL> conn hr/hr
Connected.
SQL> get by2
  1  create or replace
  2  function by2(x number) return number is
  3  begin
  4    return x*2;
  5* end;
SQL> /

Function created.
```

From other terminal I will connect to the HR session with GDB:

```
[root@rico ~]# gdb -p 10330
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-83.el6)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
```

```
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Attaching to process 10330
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle...(no debug
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libodm12.so...(no
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libodm12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libcell12.so...don
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libcell12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskgxp12.so...(n
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskgxp12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskjcx12.so...do
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskjcx12.so
Reading symbols from /lib64/librt.so.1...Reading symbols from /usr/lib/debug/lib64/
done.
Loaded symbols for /lib64/librt.so.1
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libmql1.so...(no d
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libmql1.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libipc1.so...(no d
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libipc1.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libclsra12.so...do
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libclsra12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libdbcfg12.so...(n
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libdbcfg12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libhasgen12.so...d
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libhasgen12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskgxn2.so...(no
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libskgxn2.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocr12.so...done
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocr12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocrb12.so...don
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocrb12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocrutl12.so...d
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libocrutl12.so
Reading symbols from /lib64/libaio.so.1...Reading symbols from /usr/lib/debug/lib64
done.
Loaded symbols for /lib64/libaio.so.1
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libons.so...(no de
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libons.so
Reading symbols from /lib64/libdl.so.2...Reading symbols from /usr/lib/debug/lib64/
done.
Loaded symbols for /lib64/libdl.so.2
Reading symbols from /lib64/libm.so.6...Reading symbols from /usr/lib/debug/lib64/l
done.
Loaded symbols for /lib64/libm.so.6
Reading symbols from /lib64/libpthread.so.0...Reading symbols from /usr/lib/debug/l
[Thread debugging using libthread_db enabled]
done.
Loaded symbols for /lib64/libpthread.so.0
Reading symbols from /lib64/libnsl.so.1...Reading symbols from /usr/lib/debug/lib64
done.
Loaded symbols for /lib64/libnsl.so.1
Reading symbols from /lib64/libc.so.6...Reading symbols from /usr/lib/debug/lib64/l
done.
Loaded symbols for /lib64/libc.so.6
Reading symbols from /lib64/ld-linux-x86-64.so.2...Reading symbols from /usr/lib/de
done.
Loaded symbols for /lib64/ld-linux-x86-64.so.2
Reading symbols from /usr/lib64/libnuma.so.1...Reading symbols from /usr/lib/debug/
done.
Loaded symbols for /usr/lib64/libnuma.so.1
Reading symbols from /lib64/libnss_files.so.2...Reading symbols from /usr/lib/debug
done.
Loaded symbols for /lib64/libnss_files.so.2
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libnque12.so...(no
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libnque12.so
Reading symbols from /u01/app/oracle/product/12.1.0/dbhome_1/lib/libshpkavx12.so...
Loaded symbols for /u01/app/oracle/product/12.1.0/dbhome_1/lib/libshpkavx12.so

warning: no loadable sections found in added symbol-file system-supplied DSO at 0x7
0x000000390c00e7c0 in __read_nocancel () at ../sysdeps/unix/syscall-template.S:82
82  T_PSEUDO (SYSCALL_SYMBOL, SYSCALL_NAME, SYSCALL_NARGS)
(gdb)
```

and I will set the breakpoint for plsql_run function

```
(gdb) b plsql_run
Breakpoint 1 at 0xce1c090
(gdb) info breakpoint
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000ce1c090 <plsql_run>
(gdb) command 1
Type commands for breakpoint(s) 1, one per line.
End with a line saying just "end".
>c
>end
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000ce1c090 <plsql_run>
        c
(gdb) set pagination off
(gdb) c
Continuing.
```

In the HR session, I will use my function on the table "EMPLOYEES".

```
SQL> select by2(salary) from employees;
(...output removed...)
107 rows selected.

SQL>
```

In the gdb we can see, that the breakpoint was hit 107 times – this is related to the number of rows in the table.

And what will happen if we add "PRAGMA UDF" syntax to our little function?

```
SQL> ed
Wrote file afiedt.buf

  1  create or replace
  2  function by2(x number) return number is
  3  pragma udf;
  4  begin
  5    return x*2;
  6* end;
SQL> /

Function created.

SQL> select by2(salary) from employees;
(...output removed...)
107 rows selected.
```

```
SQL>
```

The gdb shows a funny thing:

```
(gdb) info b
Num     Type           Disp Enb Address            What
2       breakpoint     keep y   0x000000000ce1c090 <plsql_run>
    breakpoint already hit 9 times
```

So why, the function was called only 9 times?
The tkprof shows something that can be interesting:

```
select by2(salary)
from
 employees


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        9      0.00       0.00          0         15          0         107
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       11      0.00       0.01          0         15          0         107
```

There was 9 fetches! Is it possible that the function was called once for each fetch? We can make sure by changing the arraysize in SQL*Plus

```
SQL> set arraysize 66
SQL> select by2(salary) from employees;
(...output removed...)
107 rows selected.

SQL>
```

The gdb shows three plsql_run executions:

```
(gdb) info b
Num     Type           Disp Enb Address            What
3       breakpoint     keep y   0x000000000ce1c090 <plsql_run>
    breakpoint already hit 3 times
        c
```

And the tkprof also:

```
select by2(salary)
from
 employees


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        3      0.00       0.00          0          9          0         107
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        5      0.00       0.00          0          9          0         107
```

So the fetch size can have actual impact on PRAGMA UDF functionality.
Let's see what will happen from inside of the PL/SQL – without throwing the results to the client:

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2 type t_nums is table of number index by pls_integer;
  3 v_nums t_nums;
  4  begin
  5 select by2(salary) bulk collect into v_nums
  6 from employees;
  7* end;
SQL> /

PL/SQL procedure successfully completed.
```

gdb shows two plsql_run hits – one for anonymous block and one for the "BY2″ function.

## CONTEXT SWITCH – PL/SQL CURSOR LOOPS AND FETCHSIZE VS OPIFCH2

Those context switches can be a real pain in the ass – there is a great article by Frits Hoogland about context switching from SQL to PL/SQL – you can read it here:
https://fritshoogland.wordpress.com/2016/01/23/plsql-context-switch/

You should also read a great blog post by Martin Widlake: Pragma UDF – Some Current Limitations
https://mwidlake.wordpress.com/2015/11/11/pragma-udf-some-current-limitations/

I also wrote a few words about reducing context switching with PRAGMA UDF:
http://blog.ora-600.pl/2015/10/29/oracle-12c-pragma-udf-the-truth/

To sum up – since there is no wait event for context switch, you can use for example perf, to verify if the function "plsql_run" has a lot of executions on your system, which can indicate that you have the context switching problem.

But this method can be used only to reveal problems with PL/SQL functions being called from SQL.

In a lots of cases there is a problem with developers not using BUIK COLLECT syntax. From Oracle 10g onwards, there is a performance feature in PL/SQL that transforms regular cursor loops to BULK COLLECTs.

Let's check the fetch size of this optimization – the function responsible for fetching the row is called opifch2.

How do we know that? Well, one of my favourite examples of showing the context switching (or fetching) problem is comparing the execution times of those two anonymous blocks:

```
SQL> declare
  2    x number;
  3  begin
  4    for i in 1..1000000 loop
  5      select 1 into x from dual;
  6    end loop;
  7  end;
  8  /
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:00:20.08

SQL> ed
Wrote file afiedt.buf

  1  declare
  2    x number;
  3  begin
  4    for i in 1..1000000 loop
  5  x:=1;
  6    end loop;
  7* end;
SQL> /

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.01
```

With perf we can see the top function calls while executing the first anonymous block:

```
# Overhead          Command      Shared Object                               Symbol
# ........  ...............  ..................  ...................................
#
    20.51%  oracle_3902_orc  [vdso]              [.] 0x0000000000000bb0
     3.28%  oracle_3902_orc  oracle              [.] opiexe
     2.46%  oracle_3902_orc  [kernel.kallsyms]   [k] __audit_syscall_entry
     2.21%  oracle_3902_orc  [kernel.kallsyms]   [k] system_call_after_swapgs
     1.83%  oracle_3902_orc  oracle              [.] opifch2
```

The function opiexe is responsible for executing some stuff – in this example: the select statement for 1 000 000 times.
The opifch2 is the function which we will be examining, since this is the function responsible for fetching.

After some investigation, I've noticed that the number of rows fetched by this function is stored in RCX registry – let's try to prove it 

I will use a simple "bulk collect limit" syntax to change the number of rows being fetched.
To display the RCX value I will use GDB with breakpoint at opifch2 function.

Preparing GDB session:

- get SPID info about your session:

```
SQL> get myspid
  1  select spid
  2  from v$session s, v$process p
  3  where s.paddr=p.addr
  4* and   s.sid=sys_context('userenv','sid')
SQL> /

SPID
------------------------
5543
```

- start GDB and prepare a breakpoint:

```
[root@rico ~]# gdb -p 5543
(gdb) b opifch2
Breakpoint 1 at 0xcb0acb0
(gdb) command 1
Type commands for breakpoint(s) 1, one per line.
End with a line saying just "end".
>p/d $rcx
>c
>end
(gdb) set pagination off
(gdb) c
Continuing.
```

Now we will execute anonymous PL/SQL block and check our RCX value for LIMIT 25, 59, 123

- LIMIT 25

PL/SQL block:

```
SQL> ;
  1  declare
  2    cursor c_sql is
  3  select salary from employees;
  4    type t_sql is table of c_sql%ROWTYPE index by pls_integer;
  5    v_sql t_sql;
  6    v_x   number;
  7  begin
  8     open c_sql;
  9     loop
 10         fetch c_sql bulk collect into v_sql limit 25;
 11         exit when c_sql%NOTFOUND;
 12         for i in v_sql.first..v_sql.last loop
 13             v_x:=v_x+v_sql(i).salary;
 14         end loop;
 15     end loop;
 16     close c_sql;
 17* end;
SQL> /
```

GDB:

```
Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$1 = 140159623635016

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$2 = 25

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$3 = 25

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$4 = 25

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$5 = 25
```

```
Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$6 = 25
```

- LIMIT 59

PL/SQL block:

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2    cursor c_sql is
  3  select salary from employees;
  4    type t_sql is table of c_sql%ROWTYPE index by pls_integer;
  5    v_sql t_sql;
  6    v_x   number;
  7  begin
  8     open c_sql;
  9     loop
 10         fetch c_sql bulk collect into v_sql limit 59;
 11         exit when c_sql%NOTFOUND;
 12         for i in v_sql.first..v_sql.last loop
 13             v_x:=v_x+v_sql(i).salary;
 14         end loop;
 15     end loop;
 16     close c_sql;
 17* end;
SQL> /

PL/SQL procedure successfully completed.
```

GDB:

```
Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$7 = 59

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$8 = 59
```

- LIMIT 123

PL/SQL block:

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2    cursor c_sql is
  3  select salary from employees;
  4    type t_sql is table of c_sql%ROWTYPE index by pls_integer;
  5    v_sql t_sql;
  6    v_x   number;
  7  begin
  8     open c_sql;
  9     loop
 10         fetch c_sql bulk collect into v_sql limit 123;
 11         exit when c_sql%NOTFOUND;
 12         for i in v_sql.first..v_sql.last loop
 13             v_x:=v_x+v_sql(i).salary;
 14         end loop;
 15     end loop;
 16     close c_sql;
```

```
 17* end;
SQL> /

PL/SQL procedure successfully completed.
```

GDB:

```
Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$9 = 123
```

Great! So now we have proved, that the number of rows fetched by opifch2 is stored at RCX.
Now we can check how many rows are being fetched by standard cursor loop optimization:

PL/SQL block:

```
SQL> ;
  1  declare
  2    cursor c_sql is
  3  select salary from employees;
  4    type t_sql is table of c_sql%ROWTYPE index by pls_integer;
  5    v_sql t_sql;
  6    v_x   number;
  7  begin
  8    for i in c_sql loop
  9     v_x:=v_x+i.salary ;
 10    end loop;
 11* end;
SQL> /

PL/SQL procedure successfully completed.
```

GDB:

```
Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$10 = 100

Breakpoint 1, 0x000000000cb0acb0 in opifch2 ()
$11 = 100
```

I made tests for different table sizes and each time it was exactly 100 rows – so this is fixed value and it doesn't depend on statistics.

I think that main problem is, that you can't figure it out based on regular wait events or statistics (or at least I don't know any).

If you check active sessions with a lot of cursor loops being executed on big tables, you will see something like this:

```
Oracle 12c - orc 19:49:45 up: 6.4h,   1 ins,    2 sn,   1 us, 2.1G mt, 66.0% db
ID %CPU LOAD %DCU   AAS  ASC  ASI  ASW  AST IOPS %FR   PGA UTPS UCPS SSRT  %DBT
 1   16    0   15   0.7    1    1    0    1    5   7   82M    0    2  41u   100

EVENT (RT)                       TOT WAITS   TIME(s)  AVG_MS  PCT    WAIT_CLASS
```

```
DB CPU                                              7          100
direct path read                          37        0       0    0      User I/O

ID   SID     SPID USR PROG S  PGA SQLID/BLOCKER OPN  E/T STA STE EVENT/*LA  W/T
 1    48     5543 HR  sqlp D 3.1M 6ucj0mtv5bf9b SEL  18s ACT CPU direct pa   3m

Enter sql_id: 6ucj0mtv5bf9b

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID: 6ucj0mtv5bf9b, child number 0
-------------------------------------------
SELECT AMOUNT_SOLD FROM SH.SALES2

Plan hash value: 497282378

-------------------------------------------------------------------------------
|   Id | Operation                          | Name       | Rows | Cost | Stale |
-------------------------------------------------------------------------------
|    0 | SELECT STATEMENT                   |            |      |  13k |       |
|    1 |  TABLE ACCESS FULL                 | SALES2     |  10M |  13k | YES   |
-------------------------------------------------------------------------------
```

Fortunately you can use operating system tools like perf – you should check out this amazing article by Luca Canali about Linux Perf Probes for Oracle Tracing:http://externaltable.blogspot.com/2016/02/linux-perf-probes-for-oracle-tracing.html

Based on Luca's findings we can create a perf probe for opifch2 function and check out fetch size for all server processes on our operating system:

```
[root@rico ~]# perf probe -x /u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle opi
Added new event:
  probe_oracle:opifch2 (on 0xc70acb0 with fetchsize=%cx)

You can now use it in all perf tools, such as:

    perf record -e probe_oracle:opifch2 -aR sleep 1
[root@rico ~]# perf record -e probe_oracle:opifch2 -aR
^C[ perf record: Woken up 149 times to write data ]
[ perf record: Captured and wrote 37.395 MB perf.data (~1633825 samples) ]

[root@rico ~]# perf script > opifch2.out
```

After recording events for some time we can process the output with a very simple AWK oneliner:

```
[root@rico ~]# awk '/oracle_/ {fsize=strtonum("0x"substr($NF,11));fsize_tab[$1 " was "f
" times)"}}' opifch2.out | sort -n -k7
Fetchsize for oracle_5857_orc was 2 (executed 1 times)
Fetchsize for oracle_5857_orc was 2147483647 (executed 1 times)
Fetchsize for oracle_5857_orc was 100 (executed 2 times)
Fetchsize for oracle_5857_orc was 140171241660488 (executed 2 times)
Fetchsize for oracle_5857_orc was 1 (executed 4 times)
Fetchsize for oracle_5857_orc was 227183808 (executed 4 times)
Fetchsize for oracle_5857_orc was 25 (executed 5 times)
Fetchsize for oracle_5755_orc was 2 (executed 22 times)
Fetchsize for oracle_5755_orc was 140058172228192 (executed 36 times)
Fetchsize for oracle_5543_orc was 100 (executed 441045 times)
```

The high number of executions for fetchsize 100 suggests that your system is using a lot of cursor loops with auto optimization – but without actual BULK COLLECT syntax.

Please notice, that this method can reveal also wrong fetch sizes from any other client – like JDBC or .NET.