Procwatcher: Script to Monitor and Examine Oracle DB and Clusterware Processes (Doc ID 459694.1)

To Bottom

In this Document

APPLIES TO:

Oracle Database Backup Service - Version N/A and later
Oracle Database Cloud Service - Version N/A and later
Oracle Database - Enterprise Edition - Version 10.2.0.2 to 12.2.0.1 [Release 10.2 to 12.2]
Oracle Database Cloud Schema Service - Version N/A and later
Oracle Database Exadata Express Cloud Service - Version N/A and later
Linux x86-64
Linux x86
HP-UX PA-RISC (64-bit)
IBM AIX on POWER Systems (64-bit)
Oracle Solaris on SPARC (64-bit)
HP-UX Itanium
Oracle Server Enterprise Edition - Version: 10.1 to 12.1

PURPOSE

Procwatcher is a tool to examine and monitor Oracle database and/or clusterware processes at an interval. The tool will collect stack traces of these processes using Oracle tools like oradebug short_stack and/or OS debuggers like pstack, gdb, dbx, or ladebug and collect SQL data if specified.

> If there are any problems with the prw.sh script or if you you have suggestions, please post a comment on this document with details.

SCOPE

This tool is for Oracle representatives and DBAs looking to troubleshoot a problem further by monitoring processes. This tool can be used in conjunction with other tools or troubleshooting methods depending on the situation.

DETAILS

```
# This script will find clusterware and/or Oracle Background processes and collect
# stack traces for debugging. It will write a file called procname_pid_date_hour.out
# for each process. If you are debugging clusterware then run this script as root.
# If you are only debugging Oracle background processes then you can run as
# root or oracle.
```

To install the script, simply download it put it in its own directory, unzip it, and give it execute permissions. Use the following link to download it:

DOWNLOAD PROCWATCHER

Alternatively, you can run Procwatcher from within the Trace File Analyzer (TFA). Procwatcher is available in the latest version of TFA (excluding 18.1.1.0.0) available for download from the following KM document:

TFA Collector - Tool for Enhanced Diagnostic Gathering Note:1513912.1

Was this document helpful?

○ Yes
○ No

Document Details

Type:	BULLETIN
Status:	PUBLISHED
Last Major Update:	Mar 7, 2019
Last Update:	Jun 10, 2021
Language:	English

Related Products

Oracle Database Backup Service
Oracle Database Cloud Service
Oracle Database - Enterprise Edition
Oracle Database - Enterprise Edition
Oracle Database Cloud Schema Service
Show More

Information Centers

Oracle Catalog: Service Request Data Collections (SRDCs) for all Products and Services [51.2]

Oracle Catalog: Information Centers and Advisors for All Products and Services [50.2]

Platform as a Service (PaaS) and Oracle Cloud Infrastructure (OCI) Information Center [2048297.2]

Index of Oracle Database Information Centers [1568043.2]

Information Center: Overview Database Server/Client Installation and Upgrade/Migration [1351022.2]

Show More

Document References

CRS Diagnostic Data Gathering: A Summary of Common tools and their Usage [783456.1]

How To Troubleshoot ORA-4031's and Shared Pool Issues With Procwatcher [1355030.1]

Process Hangs After Issuing Oradebug Short_Stack on HP Platforms [1271173.1]

Best Practices: Proactive Data Collection for Performance Issues [1477599.1]

How To Troubleshoot Database Contention With Procwatcher [1352623.1]

Show More

Recently Viewed

How To Troubleshoot Database Contention With

Note: If you had a previous version installed, stop it prior to putting the new version in place. If you installing TFA and have Procwatcher deployed, you should deinstall and re-deploy with tfactl.

If you are in a clustered environment, you can "deploy" Procwatcher with "prw.sh deploy <directory>" (or "tfactl prw deploy" in TFA) to register with the clusterware, propagate to all nodes, and start on all nodes. There is also a deinstall option to deregister from the clusterware and remove the procwatcher directory. In a clustered environment, Procwatcher files will be written to GRID_HOME/log/procwatcher unless a Procwatcher directory has been manually specified at the command line or with the PRWDIR parameter.

Requirements

- Must have /bin and /usr/bin in your $PATH
- Have your instance_name or db_name set in the oratab and/or set the $ORACLE_HOME env variable.(PRW searches the oratab for the SID it finds and if it can't find the SID in the oratab it will default to $ORACLE_HOME). Procwatcher cannot function properly if it cannot find an $ORACLE_HOME to use.
- Run Procwatcher as the oracle software owner if you are only troubleshooting homes/instances for that user. If you are troubleshooting clusterware processes (EXAMINE_CLUSTER=true or are troubleshooting for multiple oracle users) run as root.
- If you are monitoring the clusterware you must have the relevant OS debugger installed on your platform; PRW looks for:

Linux - /usr/bin/gdb
HP-UX and HP Itanium - /opt/langtools/bin/gdb64 or /usr/ccs/bin/gdb64
Sun - /usr/bin/pstack
IBM AIX - /bin/procstack or /bin/dbx
HP Tru64 - /bin/ladebug

It will use pstack on any platform where it is available besides Linux (since pstack is a wrapper script for gdb anyway).

Procwatcher Features

- Procwatcher collects stack traces for all processes defined using either oradebug short_stack or an OS debugger at a predefined interval if contentioin is found.
- PRW will generate wait chain, session wait, lock, and latch reports if problems are detected (look for pw_* reports in the PRW_DB_subdirectory).
- PRW will look for wait chains, wait events, lock, and latch contention and also dump stack traces of processes that are either waiting for non-idle wait events or waiting for or holding a lock or latch.
- PRW will dump wait chain, session wait, lock, latch, current SQL, process memory, and session history information into specific process files (look for prw_* files in the PRW_DB_subdirectory) for any processes or background processes when problems are detected.
- You can define how aggressive PRW is about getting information by setting parameters like THROTTLE, IDLECPU, and INTERVAL. You can tune these parameters to either get the most information possible or to reduce PRW's cpu impact. See below for more information about what each of these parameters does.
- If CPU usage gets too high on the machine (as defined by IDLECPU), PRW will sleep and wait for CPU utilization to go down.
- Procwatcher gets stack traces of ALL threads of a process (this is important for clusterware processes).
- The housekeeper process runs on a 5 minute loop and cleans up files older than the specified number of days (default is 7).
- If any SQL times out 90 seconds (by default) it will be disabled. At a later time the SQL can be re-tested. If the SQL times out 3 times it will be disabled for the life of Procwatcher. Any GV$ view that times out will automatically revert to the corresponding V$ view. Note that the GV$ view timeout is much lower. The logic is: it's not worth using GV$ views if they aren't fast...If oradebug shortstack is enabled and it times out or fails, the housekeeper process will re-enable shortstack if the test passes.

Disclaimer, especially if you are monitoring clusterware with EXAMINE_CLUSTER=true (default is false) or if FALL_BACK_TO_OSDEBUGGER=true (default is false): Most OS debuggers will temporarily suspend a process when attaching and dumping a stack trace. Procwatcher minimizes the amount of time that takes as much as possible. Some debuggers can also be CPU intensive. The THROTTLE,; IDLECPU, and INTERVAL parameters (see below) may need to be adjusted to suit your needs depending on how loaded the machine is and how fast it is. Note that some debuggers are faster and can get in and out of a process quicker than others. ; For example, pstack and oradebug short_stack are fast, ladebug is slower.

If you are on HP Itanium or HP-UX: Apply the fix for bug: 10158006 (or bug: 10287978 on 11.2.0.2) before monitoring the database with Procwatcher to fix a known short stack issue on HP. See Note: 1271173.1 for more information.

If you are on Solaris 10: Apply the fix for Solaris bt 6994922 (see bug: 15677306) before monitoring the database with Procwatcher.

If you are on IBM AIX please see Note: 2092006.1 and apply the relevant AIX fixes if you are planning to use EXAMINE_CLUSTER=true.

Procwatcher [1352623.1]
Troubleshooting Database Contention With V$Wait_Chains [1428210.1]
Bug 31602782 - Contention on &quot;CURSOR: PIN S WAIT ON X&quot; when PQ slave&#39;s execution plan does not match with QC [31602782.8]
Oracle Database 19c Important Recommended One-off Patches [555.1]
Bug 31602782 - Contention on "CURSOR: PIN S WAIT ON X" when PQ slave''s execution plan does not match with QC [31602782.8]
Show More

Procwatcher is Ideal for:

- Session level hangs or severe contention in the database/instance. See Note: 1352623.1
- Severe performance issues. See Note: 1352623.1
- Instance evictions and/or DRM timeouts.
- Clusterware or DB processes stuck or consuming high CPU (must set EXAMINE_CLUSTER=true and run as root for clusterware processes)
- ORA-4031 and SGA memory management issues. (Set sgamemwatch=diag or sgamemwatch=avoid4031 (not the default). See Note: 1355030.1
- ORA-4030 and DB process memory issues. (Set USE_SQL=true and process_memory=y).
- RMAN slowness/contention during a backup. (Set USE_SQL=true and rmanclient=y).

Procwatcher is Not Ideal for...

- Node evictions/reboots. In order to troubleshoot these you would have to enable Procwatcher for a process(es) that are capable of rebooting the machine. If the OS debugger suspends the processs for too long *that* could cause a reboot of the machine. I would only use Procwatcher for a node eviction/reboot if the problem was reproducing on a test system and I didn't care of the node got rebooted. Even in that case the INTERVAL would need to be set low (30) and many options would have to be turned off to get the cycle time low enough (EXAMINE_BG=false, USE_SQL=false, probably removing additional processes from the CLUSTERPROCS list).
- Non-severe database performance issues. AWR/ADDM/statspack are better options for this...
- Most installation or upgrade issues. We aren't getting data for this unless we are at a stage of the installation/upgrade where key processes are already started.

Procwatcher User Commands

To start Procwatcher:

```
./prw.sh start
```

Or if running inside of TFA:

```
tfactl prw start
```

Or if you want to start on all nodes in a clustered environment:

```
./prw.sh start all
```

Or if running inside of TFA:

```
tfactl prw start all
```

To stop Procwatcher:

```
./prw.sh stop
```

Or if running inside of TFA:

```
tfactl prw stop
```

Or if you want to stop on all nodes in a clustered environment:

```
./prw.sh stop all
```

Or if running inside of TFA:

```
tfactl prw stop all
```

To check the status of Procwatcher:

```
./prw.sh stat
```

Or if running inside of TFA:

```
tfactl prw stat
```

To package up Procwatcher files to upload to support:

```
./prw.sh pack
```

Or if running inside of TFA:

```
tfactl prw pack
```

All user syntax available:

```
./prw.sh help

Usage: prw.sh
TFA Syntax: tfactl prw <verb>

Verbs are:

deploy [directory] - Register Procwatcher in Clusterware and propagate to all nodes
start [all] - Start Procwatcher on local node, if 'all' is specified, start on all nodes
stop [all] - Stop Procwatcher on local node, if 'all' is specified, stop on all nodes
stat - Check the current status of Procwatcher
pack - Package up Procwatcher files (on all nodes) to upload to support
param - Check current Procwatcher parameters
deinstall - Deregister Procwatcher from Clusterware and remove
log [number] - See the last [number] lines of the procwatcher log file
log [runtime] - See contiuous procwatcher log file info - use Cntrl-C to break
init [directory] - Create a default prwinit.ora file
help - What you are looking at...
```

Procwatcher Parameters

Starting in Procwatcher version 12.1.14.12, these parameters are set in the prwinit.ora file in the Procwatcher directory. If you do not see a prwinit.ora file, you can generate one with "prw.sh init <directory>" or "prw.sh deploy <directory>" in a clustered environment.

```
######################### CONFIG SETTINGS #############################
# Set EXAMINE_CLUSTER variable if you want to examine clusterware processes (default is false - or set to true):
# Note that if this is set to true you must deploy/run procwatcher as root unless using oracle restart
EXAMINE_CLUSTER=false

# Set EXAMINE_BG variable if you want to examine all BG processes (default is true - or set to false):
EXAMINE_BG=true

# Set permissions on Procwatcher files and directories (default: 744):
PRWPERM=744

# Set RETENTION variable to the number of days you want to keep historical procwatcher data (default: 7)
RETENTION=7

# Warning e-mails are sent to which e-mail addresses?
# "mail" must work on the unix server
# Example: WARNINGEMAIL=john.doe@oracle.com,jane.doe@oracle.com
WARNINGEMAIL=
######################## PERFORMANCE SETTINGS ########################
# Set INVERVAL to the number of seconds between runs (default 60):
# Probably should not set below 60 if EXAMINE_CLUSTER=true
INTERVAL=60

# Set THROTTLE to the max # of stack trace sessions or SQLs to run at once (default 5 - minimum 2):
THROTTLE=5

# Set IDLECPU to the percentage of idle cpu remaining before PRW sleeps (default 3 - which means PRW will sleep if the
machine is more than 97% busy - check vmstat every 5 seconds)
IDLECPU=3

# Set SIDLIST to the list of SIDs you want to examine (default is derived - format example: RAC1|ASM1|SID3)
# If setting for multiple instances for the same DB, specify each SID - example: ASM1|ASM2|ASM3
# Default: If root is starting prw, get all sids found running at the time prw was started.
#          If another user is starting prw, get all sids found running owned by that user.
SIDLIST=
#######################################################################
```

Advanced Parameters

```
# Procwatcher log directory
# Default is $GRID_HOME/log/procwatcher if clusterware is running and this is not set
# Default is the directory where prw.sh is run if no clusterware and this is not set
# Example: PRWDIR=/home/oracle/procwatcher
PRWDIR=

# SQL Control
# Set USE_SQL variable if you want to use SQL to troubleshoot (default is true - or set to false):
USE_SQL=true
```

```
# Set to 'y' to enable SQL, 'n' to disable
sessionwait=y
lock=y
latchholder=y
gesenqueue=y
waitchains=y
rmanclient=n
process_memory=n
sqltext=y
ash=y

# SGA Memory watch (default: off).  Valid values are:
# off = no SGA memory diagnostics
# diag = collect SGA memory diagnostics
# avoid4031 = collect SGA memory diagnostics and flush the shared pool to avoid ORA-4031
#             if memory fragmentation occurs
# Note that setting sgamemwatch to 'diag' or 'avoid4031' will query x$ksmsp
# which may increase shared pool latch contention in some environments.
# Please keep this in mind and test in a test environment
# with load before using this setting in production.
sgamemwatch=off

# Levels for debugging before a flush if sgamemwatch=avoid4031 (default: 0 for both)
heapdump_level=0
lib_cache_dump_level=0

# Suspect Process Threshold (if # of suspect procs > <value> then collect BG process stacks)
# 1 = Get query and stack output if there is at least 1 suspect proc (default)
# 0 = Get all diags each cycle
suspectprocthreshold=1

# Warning Process Threshold (if # of suspect procs > <value> then issue a WARNING) default=10
warningprocthreshold=10

# Levels for debugging if warningprocthreshold is reached (default: 0 for both)
# If using this feature recommended values are (hanganalyze_level=3, systemstate_level=258)
# Flood control limits the dumps to a maximum of 3 per hour
hanganalyze_level=0
systemstate_level=0

# Cluster Process list for examination (seperated by "|"):
# Default:
"crsd.bin|evmd.bin|evmlogge|racgimon|racge|racgmain|racgons.b|ohasd.b|oraagent|oraroota|gipcd.b|mdnsd.b|gpnpd.b|gnsd.
octssd.b|ons -d|tnslsnr"
# - The processes oprocd, cssdagent, and cssdmonitor are intentionally left off the list because of high reboot danger.
# - The ocssd.bin process is off the list due to moderate reboot danger.  Only add this if your css misscount is the
# - default or higher, your machine is not highly loaded, and you are aware of the tradeoff.
CLUSTERPROCS="crsd.bin|evmd.bin|evmlogge|racgimon|racge|racgmain|racgons.b|ohasd.b|oraagent|oraroota|gipcd.b|mdnsd
gnsd.bi|diskmon|octssd.b|ons -d|tnslsnr"

# DB Process list for examination (seperated by "|"):
# Default: "_dbw|_smon|_pmon|_lgwr|_lmd|_lms|_lck|_lmon|_ckpt|_arc|_rvwr|_gmon|_lmhb|_rms0"
# - To examine ALL oracle DB and ASM processes on the machine, set BGPROCS="ora|asm" (not typically recommended)
BGPROCS="_dbw|_smon|_pmon|_lgwr|_lmd|_lms|_lck|_lmon|_ckpt|_arc|_rvwr|_gmon|_lmhb|_rms0"

# Set to 'y' to enable gv$views, set to 'n' to disable gv$ views
# (makes queries a little faster in RAC but can't see other instances in reports)
# Default is derived based on if waitchains is used
use_gv=

# Set to 'y' to get pmap data for clusterware processes.
# Only available on Linux and Solaris
use_pmap=n

# DB Versions enabled, set to 'y' or 'n' (this will override the SIDLIST setting)
VERSION_10_1=y
VERSION_10_2=y
VERSION_11_1=y
VERSION_11_2=y

# Should we fall back to an OS debugger if oradebug short_stack fails?
# OS debuggers are less safe per bug 6859515 so default is false (or set to true)
FALL_BACK_TO_OSDEBUGGER=false

# Number of oradebug shortstacks to get on each pass
# Will automatically lower if stacks are taking too long
STACKCOUNT=3

# Point this to a custom .sql file for Procwatcher to capture every cycle.
# Don't use big or long running SQL.  The .sql file must be executable.
# Only 1 SQL per file.
# Example: CUSTOMSQL1=/home/oracle/test.sql
CUSTOMSQL1=
CUSTOMSQL2=
CUSTOMSQL3=
```

REFERENCES

NOTE:783456.1 - CRS Diagnostic Data Gathering: A Summary of Common tools and their Usage
NOTE:1355030.1 - How To Troubleshoot ORA-4031's and Shared Pool Issues With Procwatcher
NOTE:1271173.1 - Process Hangs After Issuing Oradebug Short_Stack on HP Platforms

NOTE:1477599.1 - Best Practices: Proactive Data Collection for Performance Issues
NOTE:1352623.1 - How To Troubleshoot Database Contention With Procwatcher
NOTE:452358.1 - How to Collect Diagnostics for Database Hanging Issues

NOTE:1594347.1 - RAC and DB Support Tools Bundle
NOTE:430473.1 - ORA-4031 Common Analysis/Diagnostic Scripts

NOTE:1353073.1 - Exadata Diagnostic Collection Guide

NOTE:1389167.1 - Get Proactive with Oracle Database
NOTE:559339.1 - Diagnostic Tools Catalog
NOTE:396940.1 - Troubleshooting and Diagnosing ORA-4031 Error [Video]
NOTE:1513912.1 - TFA Collector - TFA with Database Support Tools Bundle
NOTE:1428210.1 - Troubleshooting Database Contention With V$Wait_Chains

Didn't find what you are looking for? Ask in Community...

Attachments

Procwatcher Version 19.8.20.9.0 (39.64 KB)

Related

Products

- Oracle Cloud > Oracle Platform Cloud > Oracle Database Backup Service > Oracle Database Backup Service
- Oracle Cloud > Oracle Platform Cloud > Oracle Database Cloud Service > Oracle Database Cloud Service
- Oracle Database Products > Oracle Database Suite > Oracle Database > Oracle Database - Enterprise Edition > RDBMS > Locking Issues
- Oracle Database Products > Oracle Database Suite > Oracle Database > Oracle Database - Enterprise Edition > Real Application Cluster
- Oracle Cloud > Oracle Platform Cloud > Oracle Database Cloud Service > Oracle Database Cloud Schema Service
- Oracle Cloud > Oracle Platform Cloud > Oracle Database Cloud Service > Oracle Database Exadata Express Cloud Service
- Oracle Cloud > Oracle Infrastructure Cloud > Oracle Cloud at Customer > Gen 1 Exadata Cloud at Customer (Oracle Exadata Database Cloud Machine)
- Oracle Cloud > Oracle Platform Cloud > Oracle Cloud Infrastructure - Database Service > Oracle Cloud Infrastructure - Database Service
- Oracle Cloud > Oracle Platform Cloud > Oracle Database Cloud Exadata Service > Oracle Database Cloud Exadata Service

Errors

ORA-4030; ORA-4031; SQL1; SQL2; SQL3

Translations

- English Source
- Japanese 日本語

Back to Top

| Legal Notices and Terms of Use | Privacy Statement