CoE logo

OSWbb

OS Watcher Black Box User's Guide

Carl Davis
Center of Expertise
February 1, 2012

```
@Please Note:  OSW has been renamed to OSWbb (OSWatcher Black Box) to prevent
@confusion as there are several tools now within Oracle that share this same name.
```

OSWbb now provides an analysis tool OSWbba which analyzes the log files produced by OSWbb. This tool allows OSWbb to be self-analyzing. This tool also provides a graphing capability to graph the data and to produce a htmp profile. See the "Graphing the Output" section below.

To collect database metrics in addition to OS metrics consider running LTOM. To see an example of your system profiled with LTOM click here.

Contents

Introduction

OS Watcher Black Box (OSWbb) is a collection of UNIX shell scripts intended to collect and archive operating system and network metrics to aid support in diagnosing performance issues. OSWbb operates as a set of background processes on the server and gathers OS data on a regular basis, invoking such Unix utilities as vmstat, netstat and iostat. OSWbb can be downloaded from this note. OSWbb is also included in the RAC-DDT script file, but is not installed by RAC-DDT. For more information on RAC-DDT see <>. OSWbb is installed on each node where data is to be collected. Installation instructions for OSWbb are provided in this user guide.

Back to Contents

Overview

OSWbb consists of a series of shell scripts. OSWatcher.sh is the main controlling executive, which spawns individual shell processes to collect specific kinds of data, using Unix operating system diagnostic utilities. Control is passed to individually spawned operating system data collector processes, which in turn collect specific data, timestamp the data output, and append the data to pre-generated and named files. Each data collector will have its own file, created and named by the File Manager process.

Data collection intervals are configurable by the user, but will be uniform for all data collector processes for a single instance of the OSWbb tool. For example, if OSWbb is configured to collect data once per minute, each spawned data collector process will generate output for its respective metric, write data to its corresponding data file, then sleep for one minute (or other configured interval) and repeat. Because we are collecting data every minute, the files generated by each spawned processes will contain 60 entries, one for each minute during the previous hour. Each file will contain, at most, one hour of data. At the end of each hour, File Manager will wake up and copy the existing current hour file to an archive location, then create a new current hour file.

The File Manager ensures only the last *N* hours of information are retained, where *N* is a configurable integer defaulting to 48. File Manager will wake up once per hour to delete files older than *N* hours. At any time, the entire output file set will consist of one current hour file, plus *N* archive files for each data collector process.

stopOSWbb.sh will terminate all processes associated with OSWbb, and is the normal, graceful mechanism for stopping the tool's operation.

OSWbb invokes these distinct operating system utilities, each as a distinct background process, as data collectors. These utilities will be supported, or their equivalents, as available for each supported target platform.

- ps
- top
- mpstat
- iostat
- netstat
- traceroute
- vmstat

Back to Contents

Supported Platforms

OSWbb is certified to run on the following platforms:

- AIX
- Tru64
- Solaris
- HP-UX
- Linux

Back to Contents

Gathering Diagnostic Data

Back to Contents

Installing OSWbb

OSWbb needs to be installed on each node, one installation per node. OSWbb should be installed manually by using the following procedure:

> **NOTE**: OSWbb is available through MOS and can be downloaded as a tar file. The user then copies the file oswbb.tar to the directory where OSWbb is to be installed and issues the following commands.

```
tar xvf oswbb.tar
```

A directory named oswbb is created which houses all the files associated with OSWbb. OSWbb is now installed.

Back to Contents

Uninstalling OSWbb

To de-install OSWbb issue the following command on the oswbb directory.

```
rm -rf oswbb
```

Back to Contents

Setting up OSWbb

Once OSWbb is installed, scripts have been provided to start and stop the OSWbb utility. When OSWbb is started for the first time it creates the archive subdirectory. The archive directory contains 7 subdirectories, one for each data collector. Data collectors exist for top, vmstat, iostat, mpstat, netstat, ps and an optional collector for tracing private networks. To turn on data collection for private networks the user must create an executable file in the oswbb directory named private.net. An example of what this file should look like is named Example private.net with samples for each operating system: solaris, linux, aix, hp, etc. in the oswbb directory. This file can be edited and renamed private.net or a new file named private.net can be created. This file contains entries for running the traceroute command to verify RAC private networks.

Example private.net entry on Solaris:

```
traceroute -r -F node1
traceroute -r -F node2
```

Where node1 and node2 are 2 nodes in addition to the hostnode of a 3 node RAC cluster. If the file private.net does not exist or is not executable then no data will be collected and stored under the oswprvtnet directory.

OSWbb will need access to the OS utilities: *top, vmstat, iostat, mpstat, netstat,* and *traceroute.* These OS utilities need to be install on the system prior to running OSWbb. Execute permission on these utilities need to be granted to the user of OSWbb.

Back to Contents

Starting OSWbb

To start the OSWbb utility execute the startOSWbb.sh shell script from the directory where OSWbb was installed. This script has 2 arguments which control the frequency that data is collected and the number of hour's worth of data to archive.

ARG1 = snapshot interval in seconds.
ARG2 = the number of hours of archive data to store.

If you do not enter any arguments the script runs with default values of 30 and 48 meaning collect data every 30 seconds and store the last 48 hours of data in archive files.

Example 1:

```
./startOSWbb.sh 60 10
```

This would start the tool and collect data at 60 second intervals and log the last 10 hours of data to archive files.

Example 2:

```
./startOSWbb.sh
```

NOTE: This would use the default values of 30, 48 and collect data at 30 second intervals and log the last 48 hours of data to archive files.

Example 3:

```
nohup ./startOSWbb.sh 60 10 &
```

This would start the tool, put the process in the background, enable to the tool to continue running after the session has been terminated, collect data at 60 second intervals, and log the last 10 hours of data to archive files.

Back to Contents

Stopping OSWbb

To stop the OSWbb utility execute the stopOSWbb.sh command from the directory where OSWbb was installed. This terminates all the processes associated with the tool.

Example:

```
./stopOSWbb.sh
```

Back to Contents

Diagnostic Data Output

As stated above, when OSWbb is started for the first time it creates the archive subdirectory under the OSWbb installation directory. The archive directory contains 7 subdirectories, one for each data collector. These directories are named oswiostat, oswmpstat, oswnetstat, oswprvtnet, oswps, oswtop, and oswvmstat. One file per hour will be generated in each of the 7 OS utility subdirectories with the exception of oswprvtnet which is dependent on having private networks tracing configured. A new file is created at the top of each hour during the time that OSWbb is

running. The file will be in the following format:

```
<node_name>_<OS_utility>_YY.MM.DD.HH24.dat
```

Details about each type of data file can be viewed by clicking on the below links:

oswiostat

<node_name>_iostat_YY.MM.DD:HH24.dat

These files will contain output from the 'iostat' command that is obtained and archive by OSWatcher Black Box at specified intervals. These files will only exist if 'iostat' is installed on the OS and if the OSWbb user has privileges to run the utility.

The iostat command is used for monitoring system input/output device loading by observing the time the physical disks are active in relation to their average transfer rates. This information can be used to change system configuration to better balance the input/output load between physical disks and adapters.

The iostat utility is fairly standard across UNIX platforms, but really on useful for those platforms that support extended disk statistics: AIX, Solaris and Linux. Also each platform will have a slightly different version of the iostat utility. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the iostat utility at the specified interval and stores the data in the oswiostat subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the iostat output. Notice there are 3 entries for each timestamp. You should always ignore the first entry as this entry is always invalid. The second and third entry will be valid but the second entry will be 1 sec later than the timestamp and the third entry will be 2 seconds later than the timestamp.

Sample iostat file produced by OSWbb

```
extended device statistics
    r/s    w/s   kr/s   kw/s   wait   actv wsvc_t asvc_t  %w  %b      device
    0.0    0.3    0.0    2.1    0.0    0.0    3.4    0.8   0   0      c0t0d0
    0.0    2.1    0.1   12.9    0.0    0.0    0.6    0.4   0   0      c0t2d0
    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0   0         fd0
    2.9    1.2  240.8    1.5    0.0    0.1    0.0   13.3   0   5      c1t0d0
    1.1    0.8   18.0    8.8    0.0    0.0    0.1    5.9   0   1      c1t1d0
    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0   0      c0t1d0
```

Field Descriptions

The iostat output contains summary information for all devices.

Field	Description
r/s	Shows the number of reads/second
w/s	Shows the number of writes/second
kr/s	Shows the number of kilobytes read/second
kw/s	Shows the number of kilobytes written/second

wait	Average number of transactions waiting for service (queue length)
actv	Average number of transactions actively being serviced
wsvc_t	Average service time in wait queue, in milliseconds
asvc_t	Average service time of active transactions, in milliseconds
%w	Percent of time there are transactions waiting for service
%b	Percent of time the disk is busy
device	Device name

What to look for

- Average service times greater than 20msec for long duration.
- High average wait times.

Back to Contents

oswmpstat

<node_name>_mpstat_YY.MM.DD:HH24.dat

These files will contain output from the 'mpstat' command that is obtained and archive by OSWatcher Black Box at specified intervals. These files will only exist if 'mpstat' is installed on the OS and if the OSWbb user has privileges to run the utility.

The mpstat command collects and displays performance statistics for all logical CPUs in the system.

The mpstat utility is fairly standard across UNIX platforms. Each platform will have a slightly different version of the mpstat utility. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the mpstat utility at the specified interval and stores the data in the oswmpstat subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the mpstat output. Notice there are 3 entries for each timestamp. You should always ignore the first entry as this entry is always invalid. The second and third entry will be valid but the second entry will be 1 sec later than the timestamp and the third entry will be 2 seconds later than the timestamp.

Sample mpstat file produced by OSWbb

```
***Fri Jan 28 12:50:36 EST 2005
 CPU  minf mjf  xcal  intr  ithr csw  icsw  migr  smtx srw  syscl usr sys wt idl
   0     0   0     0   483   383 118     1     0     0   0     64   0   0  0 100
   0  1268   0     0   486   382 414    42     0     0   0   2902   8  24  0  68
   0     4   0     0   479   379 144     3     0     0   0     96   0   0  0 100
```

Field Descriptions

Field	Description
cpu	Processor ID
minf	Minor faults
mif	Major Faults
xcal	Processor cross-calls (when one CPU wakes up another by interrupting it).
intr	Interrupts
ithr	Interrupts as threads (except clock)

csw	Context switches
icsw	Involuntary context switches
migr	Thread migrations to another processor
smtx	Number of times a CPU failed to obtain a mutex
srw	Number of times a CPU failed to obtain a read/write lock on the first try
syscl	Number of system calls
usr	Percentage of CPU cycles spent on user processes
sys	Percentage of CPU cycles spent on system processes
wt	Percentage of CPU cycles spent waiting on event
idl	Percentage of unused CPU cycles or idle time when the CPU is basically doing nothing

What to look for

- Involuntary context switches (this is probably the more relevant statistic when examining performance issues.)
- Number of times a CPU failed to obtain a mutex. Values consistently greater than 200 per CPU causes system time to increase.
- xcal is very important, show processor migration

Back to Contents

oswnetstat

<node_name>_netstat_YY.MM.DD:HH24.dat

These files will contain output from the 'netstat' command that is obtained and archive by OSWatcher Black Box at specified intervals. These files will only exist if 'netstat' is installed on the OS and if the OSWbb user has privileges to run the utility.

The **netstat** command displays current TCP/IP network connections and protocol statistics.

The netstat utility is standard across UNIX platforms. Each platform will have a slightly different version of the netstat utility. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the netstat utility at the specified interval and stores the data in the oswnetstat subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the netstat output. Notice there are 3 entries for each timestamp. You should always ignore the first entry as this entry is always invalid. The second and third entry will be valid but the second entry will be 1 sec later than the timestamp and the third entry will be 2 seconds later than the timestamp.

The netstat utility has many command line flags, and the most commonly used to troubleshoot RAC is "ia(n)" for the interface level output and "s" for the protocol level statistics. The following are examples for the two different command parameters.

The command line options "-ain" have these effects:

Option	Description
-a	The command output will use the logical names of the interface. It will also report the name of the IP address found through normal IP address resolution methods.
-i	This triggers the Interface specific statistics, the columns of which are outlined in table [bla-KR]

-n	This causes the output to use IP addresses instead of the resolved names

Example netstat file produced by OSWbb:

Sample netstat file produced by OSWbb

```
***Fri Jan 28 12:50:36 EST 2005
 Name  Mtu      Net/Dest       Address  Ipkts Ierrs  Opkts Oerrs Collis Queue
  lo0 8232    127.0.0.0     127.0.0.1 296065     0 296065     0      0     0
 eri0 1500  138.1.140.0  138.1.140.96            0 176244     2 191951     0
RAWIP
       rawipInDatagrams        =         0 rawipInErrors          =         0
       rawipInCksumErrs        =         0 rawipOutDatagrams      =         0
       rawipOutErrors          =         0
UDP
       udpInDatagrams          =    295719 udpInErrors            =         0
       udpOutDatagrams         =    295671 udpOutErrors           =         0
TCP
       tcpRtoAlgorithm         =         4 tcpRtoMin              =       400
       tcpRtoMax               =     60000 tcpMaxConn             =        -1
       tcpActiveOpens          =        27 tcpPassiveOpens        =        21
       tcpAttemptFails         =         6 tcpEstabResets         =         0
       tcpCurrEstab            =        15 tcpOutSegs             =       691
       tcpOutDataSegs          =       479 tcpOutDataBytes        =     43028
       tcpRetransSegs          =         0 tcpRetransBytes        =         0
       tcpOutAck               =       212 tcpOutAckDelayed       =        83
       tcpOutUrg               =         0 tcpOutWinUpdate        =         0
       tcpOutWinProbe          =         0 tcpOutControl          =        85
       tcpOutRsts              =        10 tcpOutFastRetrans
       tcpInSegs               =       915                        =         0
       tcpInAckSegs            =       489 tcpInAckBytes          =     43023
       tcpInDupAck             =        42 tcpInAckUnsent         =         0
       tcpInInorderSegs        =       477 tcpInInorderBytes      =     40640
       tcpInUnorderSegs        =         0 tcpInUnorderBytes      =         0
       tcpInDupSegs            =         0 tcpInDupBytes          =         0
       tcpInPartDupSegs        =         0 tcpInPartDupBytes      =         0
       tcpInPastWinSegs        =         0 tcpInPastWinBytes      =         0
       tcpInWinProbe           =         0 tcpInWinUpdate         =         0
       tcpInClosed             =         0 tcpRttNoUpdate         =         0
       tcpRttUpdate            =       462 tcpTimRetrans          =         0
       tcpTimRetransDrop       =         0 tcpTimKeepalive        =        80
       tcpTimKeepaliveProbe    =         0 tcpTimKeepaliveDrop    =         0
       tcpListenDrop           =         0 tcpListenDropQ0        =         0
       tcpHalfOpenDrop         =         0 tcpOutSackRetrans      =         0
IPv4
       ipForwarding            =         2 ipDefaultTTL           =       255
       ipInReceives            =  17858585 ipInHdrErrors          =         0
       ipInAddrErrors          =         0 ipInCksumErrs          =         0
       ipForwDatagrams         =         0 ipForwProhibits        =         0
       ipInUnknownProtos       =         0 ipInDiscards           =         0
       ipInDelivers            =    296623 ipOutRequests          =  17624403
       ipOutDiscards           =         0 ipOutNoRoutes          =       827
       ipReasmTimeout          =        60 ipReasmReqds           =         0
       ipReasmOKs              =         0 ipReasmFails           =         0
       ipReasmDuplicates       =         0 ipReasmPartDups        =         0
       ipFragOKs               =         0 ipFragFails            =         0
       ipFragCreates           =         0 ipRoutingDiscards      =         0
       tcpInErrs               =         0 udpNoPorts             =    225722
       udpInCksumErrs          =         0 udpInOverflows         =         0
       rawipInOverflows        =         0 ipsecInSucceeded       =         0
       ipsecInFailed           =         0 ipInIPv6               =         0
       ipOutIPv6               =         0 ipOutSwitchIPv6        =         5
IPv6
       ipv6Forwarding          =         2 ipv6DefaultHopLimit    =       255
       ipv6InReceives          =         0 ipv6InHdrErrors        =         0
       ipv6InTooBigErrors      =         0 ipv6InNoRoutes         =         0
       ipv6InAddrErrors        =         0 ipv6InUnknownProtos    =         0
```

```
       ipv6InTruncatedPkts    =         0 ipv6InDiscards         =         0
       ipv6InDelivers         =         0 ipv6OutForwDatagrams   =         0
       ipv6OutRequests        =         0 ipv6OutDiscards        =         0
       ipv6OutNoRoutes        =         0 ipv6OutFragOKs         =         0
       ipv6OutFragFails       =         0 ipv6OutFragCreates     =         0
       ipv6ReasmReqds         =         0 ipv6ReasmOKs           =         0
       ipv6ReasmFails         =         0 ipv6InMcastPkts        =         0
       ipv6OutMcastPkts       =         0 ipv6ReasmDuplicates    =         0
       ipv6ReasmPartDups      =         0 ipv6ForwProhibits      =         0
       udpInCksumErrs         =         0 udpInOverflows         =         0
       rawipInOverflows       =         0 ipv6InIPv4             =         0
       ipv6OutIPv4            =         0 ipv6OutSwitchIPv4      =         0
ICMPv4
       icmpInMsgs             =  17624914 icmpInErrors           =         0
       icmpInCksumErrs        =         0 icmpInUnknowns         =         0
       icmpInDestUnreachs     =        72 icmpInTimeExcds        =         0
       icmpInParmProbs        =         0 icmpInSrcQuenchs       =         0
       icmpInRedirects        =         0 icmpInBadRedirects     =         0
       icmpInEchos            =  17624842 icmpInEchoReps         =         0
       icmpInTimestamps       =         0 icmpInTimestampReps    =         0
       icmpInAddrMasks        =         0 icmpInAddrMaskReps     =         0
       icmpInFragNeeded       =         0 icmpOutMsgs            =  17624920
       icmpOutDrops           =    225716 icmpOutErrors          =         0
       icmpOutDestUnreachs    =        78 icmpOutTimeExcds       =         0
       icmpOutParmProbs       =         0 icmpOutSrcQuenchs      =         0
       icmpOutRedirects       =         0 icmpOutEchos           =         0
       icmpOutEchoReps        =  17624842 icmpOutTimestamps      =         0
       icmpOutTimestampReps   =         0 icmpOutAddrMasks       =         0
       icmpOutAddrMaskReps    =         0 icmpOutFragNeeded      =         0
       icmpInOverflows        =         0
ICMPv6
       icmp6InMsgs            =         0 icmp6InErrors          =         0
       icmp6InDestUnreachs    =         0 icmp6InAdminProhibs    =         0
       icmp6InTimeExcds       =         0 icmp6InParmProblems    =         0
       icmp6InPktTooBigs      =         0 icmp6InEchos           =         0
       icmp6InEchoReplies     =         0 icmp6InRouterSols      =         0
       icmp6InRouterAds       =         0 icmp6InNeighborSols    =         0
       icmp6InNeighborAds     =         0 icmp6InRedirects       =         0
       icmp6InBadRedirects    =         0 icmp6InGroupQueries    =         0
       icmp6InGroupResps      =         0 icmp6InGroupReds       =         0
       icmp6InOverflows       =         0
       icmp6OutMsgs           =         0 icmp6OutErrors         =         0
       icmp6OutDestUnreachs   =         0 icmp6OutAdminProhibs   =         0
       icmp6OutTimeExcds      =         0 icmp6OutParmProblems   =         0
       icmp6OutPktTooBigs     =         0 icmp6OutEchos          =         0
       icmp6OutEchoReplies    =         0 icmp6OutRouterSols     =         0
       icmp6OutRouterAds      =         0 icmp6OutNeighborSols   =         0
       icmp6OutNeighborAds    =         0 icmp6OutRedirects      =         0
       icmp6OutGroupQueries   =         0 icmp6OutGroupResps     =         0
       icmp6OutGroupReds      =         0
 IGMP:
      2490 messages received
         0 messages received with too few bytes
         0 messages received with bad checksum
      2490 membership queries received
         0 membership queries received with invalid field(s)
         0 membership reports received
         0 membership reports received with invalid field(s)
         0 membership reports received for groups to which we belong

         0 membership reports sent
```

Field Descriptions:

The netstat output produced by OSWbb contains 2 sections. The first section contains information about all the network interfaces. The second section contains information about per-protocol statistics.

Section 1: Netstat -ain

Field	Description
name	Device name of interface
Mtu	Maximum transmission unit
Net	Network Segment Address
address	Network address of the device
ipkts	Input packets
Ierrs	Input errors
opkts	Output Packets
Oerrs	Output errors
collis	Collisions
queue	Number in the Queue

Section 2: Protocol Statistics

The per-protocol statistics can be divided into several categories:

- RAWIP (raw IP) packets
- TCP packets
- IPv4 packets
- ICMPv4 packets
- IPv6 packets
- ICMPv6 packets
- UDP packets
- IGMP packet

Each protocol type has a specific set of measures associated with it. Network analysis requires evaluation of these measurements on an individual level and all together to examine the overall health of the network communications.

The TCP protocol is used the most in Oracle database and applications. Some implementations for RAC use UDP for the interconnect protocol instead of TCP. The statistics cannot be divided up on a per-interface basis, so these should be compared to the "-i" statistics above.

What to look for:

Section 1

The information in Section 1 will help diagnose network problems when there is connectivity but response is slow.

Values to look at:

- Collisions (Collis)
- Output packets (Opkts)
- Input errors (Ierrs)
- Input packets (Ipkts)

The above values will give information to workout network collision rates as follows:

Network collision rate = Output collision / Output packets

For a switched network, the collisions should be 0.1 percent or less (see the Cisco web site as a reference) of the output packets. Excessive collisions could lead to the switch port the interface is plugged into to segment, or pull itself off-line, amongst other switch-related issues.

For the input error statistics:

Input Error Rate = Ierrs / Ipkts.

If the input error rate is high (over 0.25 percent), the host is excessively dropping packets. This could mean there is a mismatch of the duplex or speed settings of the interface card and switch. It could also imply a failed patch cable.

If ierrs or oerrs show an excessive amount of errors, more information can be found by examination of the netstat -s output.

For Sun systems, further information about a specific interface can be found by using the "-k" option for netstat. The output will give fuller statistics for the device, but this option is not mentioned in the netstat man page. More information can be found at http://sunsolve.sun.com.

Section 2

The information in Section 2 contains the protocol statistics.

Many performance problems associated with the network involve the retransmission of the TCP packets. For retransmission rate calculations click here.

To find the segment retransmission rate:

%segment-retrans=(tcpRetransSegs / tcpOutDataSegs) * 100

To find the byte retransmission rate:

%byte-retrans = (tcpRetransBytes / tcpOutDataBytes) * 100

Most network analyzers report TCP retransmissions as segments (frames) and not in bytes.

Back to Contents

oswprvtnet

<node_name>_prvtnet_YY.MM.DD:HH24.dat

These files will contain output from the 'prvtnet' command that is obtained and archived by OSWatcher Black Box at specified intervals. These files will only exist if 'prvtnet' is installed on the OS and if the OSWbb user has privileges to run the utility.

Information about the status of RAC private networks should be collected. This requires the user to manually add entries for these private networks into the private.net file located in the base oswbb directory. Instructions on how to do this are contained in the README file.

OSWbb uses the traceroute command to obtain the status of these private networks. Each operating system uses slightly different arguments to the traceroute command. Examples of the syntax to use for each operating system are contained in the sample Example private.net file located in the base oswbb directory. This will result in the output appearing differently across UNIX platforms. OSWbb runs the private.net file at the specified interval and stores the data in the oswprvtnet subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the top output.

Sample file produced by OSWbb

```
***Fri Jan 28 12:50:36 EST 2005

traceroute to celdecclu2.us.oracle.com (138.2.71.112): 1-30 hops
(initial packetsize = 1500)
  1  celdecclu2.us.oracle.com (138.2.71.112) 1.95ms  2.92 ms 1.95 ms
```

What to Look For

- Example 1: Interface is up and responding:

```
traceroute to X.X.X.X, (X.X.X.X) 30 hops max, 1492 byte packets
1 X.X.X.X 1.015 ms 0.766 ms 0.755 ms
```

- Example 2: Target interface is not on a directly connected network, so validate that the address is correct or the switch it is plugged in is on the same VLAN (or other issue):

```
traceroute to X.X.X.X, (X.X.X.X) 30 hops max, 40 byte packets
traceroute: host X.X.X.X is not on a directly-attached network
```

- Example 3: Network is unreachable:

```
traceroute to X.X.X.X, (X.X.X.X) 30 hops max, 40 byte packets
Network is unreachable
```

Back to Contents

oswps

<node_name>_ps_YY.MM.DD:HH24.dat

These files will contain output from the 'ps' command that is obtained and archive by OSWatcher Black Box at specified intervals. These files will only exist if 'ps' is installed on the OS and if the OSWbb user has privileges to run the utility.

The ps (process state) command list all the processes currently running on the system and provides information about CPU consumption, process state, priority of the process, etc. The ps command has a number of options to control which processes are displayed, and how the output is formatted. OSWbb runs the ps command with the -elf option.

The ps command is fairly standard across UNIX platforms Each platform will have a slightly different version of the ps utility. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the ps command at the specified interval and stores the data in the oswps subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the ps output.

Sample ps file produced by OSWbb

```
***Wed Feb 2 09:26:54 EST 2005
F  S UID       PID  PPID  C PRI  NI ADDR  SZ   WCHAN   STIME    TTY  TIME  CMD
19 T root      0    0     0 0    SY ?     0            Jan 31   ?    0:13  sched
8  S root      1    0     0 41   20 ?     107  ?       Jan 31   ?    0:00  /etc
19 S root      2    0     0 0    SY ?     0    ?       Jan 31   ?    0:00  page
19 S root      3    0     0 0    SY ?     0    ?       Jan 31   ?    0:50  fsflu
8  S root      355  1     0 41   20 ?     232  ?       Jan 31   ?    0:00  /usr/
8  S root      297  296   0 41   20 ?     379  ?       Jan 31   ?    0:00  htt_s
8  S cedavis   391  381   0 89   20 ?     301  ?       Jan 31   ?    0:00  /usr/
```

Field Descriptions

Field	Description
f	Flags s State of the process
uid	The effective user ID number of the process

pid	The process ID of the process
ppid	The process ID of the parent process.
d	Processor utilization for scheduling (obsolete).
pri	The priority of the process.
ni	Nice value, used in priority computation.
addr	The memory address of the process.
sz	The total size of the process in virtual memory, including all mapped files and devices, in pages.
wchan	The address of an event for which the process is sleeping (if blank, the process is running).
stime	The starting time of the process, given in hours, minutes, and seconds.
tty	The controlling terminal for the process (the message ?, is printed when there is no controlling terminal).
time	The cumulative execution time for the process.
cmd	The command name process is executing.

What to look for

- The information in the ps command will primarily be used as supporting information for RAC diagnostics. If for example, the status of a process prior to a system crash may be important for root cause analysis. The amount of memory a process is consuming is another example of how this data can be used.

Back to Contents

oswtop

<node_name>_top_YY.MM.DD:HH24.dat

These files will contain output from the 'top' command that is obtained and archive by OSWatcher at specified intervals. These files will only exist if 'top' is installed on the OS and if the OSWbb user has privileges to run the utility.

Top is a program that will give continual reports about the state of the system, including a list of the top CPU using processes. Top has three primary design goals:

- provide an accurate snapshot of the system and process state,
- not be one of the top processes itself,
- be as portable as possible.

Each operating system uses a different version of the UNIX utility top. This will result in the top output appearing differently across UNIX platforms. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the top utility at the specified interval and stores the data in the oswtop subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the top output.

Sample top file produced by OSWbb

```
***Fri Jan 28 12:50:36 EST 2005
load averages: 0.11, 0.07, 0.06 12:50:36
136 processes: 133 sleeping, 2 running, 1 on cpu

Memory: 2048M real, 1061M free, 542M swap in use, 1605M swap free
PID   USERNAME  THR PRI NICE SIZE  RES   STATE    TIME    CPU   COMMAND
704   cedavis   16  49  0      346M  276M  sleep  222:33 3.51% java
362   root      1   59  0       34M   75M  sleep  11:49  0.21% Xsun
20675 cedavis   1   0   0     1584K 1064K  cpu    0:00   19%   top
20640 cedavis   1   0   0     1904K 1240K  sleep  0:00   0.14% OSWatcher.sh
20657 cedavis   1   20  0     1904K 1240K  sleep  0:00   0.14% oswsub.sh
```

```
16881 cedavis   1   59  0      199M  159K  sleep 23:04  0.10% oracle
20671 cedavis   1   0   0     1904K 1240K  run    0:00   0.09% oswsub.sh
20653 cedavis   1   0   0     1904K 1240K  sleep  0:00   0.09% OSWatcherFM.sh
20665 cedavis   1   0   0     1904K 1240K  sleep  0:00   0.09% oswsub.sh
20672 cedavis   1   0   0     1264K 1031K  sleep  0:00   0.09% iostat
20659 cedavis   1   10  0     1904K 1240K  sleep  0:00   0.09% oswsub.sh
20661 cedavis   1   30  0     1096K  880Ksleep    0:00   0.09% vmstat
20668 cedavis   1   0   0     1904K 1240Krun      0:00   0.05% oswsub.sh
20674 cedavis   1   0   0      968K  624K  sleep  0:00   0.05% sleep
20663 cedavis   1   20  0     1080K  864Ksleep    0:00   0.05% mpstat
```

Field Descriptions

load averages: 0.11, 0.07, 0.06 12:50:36

This line displays the load averages over the last 1, 5 and 15 minutes as well as the system time. This is quite handy as top basically includes a timestamp along with the data capture.

Load average is defined as the average number of processes in the run queue. A runnable Unix process is one that is available right now to consume CPU resources and is not blocked on I/O or on a system call. The higher the load average, the more work your machine is doing.

The three numbers are the average of the depth of the run queue over the last 1, 5, and 15 minutes. In this example we can see that .11 processes were on the run queue on average over the last minute, .07 processes on average on the run queue over the last 5 minutes, etc. It is important to determine what the average load of the system is through benchmarking and then look for deviations. A dramatic rise in the load average can indicate a serious performance problem.

136 processes: 133 sleeping, 2 running, 1 on cpu

This line displays the total number of processes running at the time of the last update. It also indicates how many Unix processes exist, how many are sleeping (blocked on I/O or a system call), how many are stopped (someone in a shell has suspended it), and how many are actually assigned to a CPU. This last number will not be greater than the number of processors on the machine, and the value should also correlate to the machine's load average provided the load average is less than the number of CPUs. Like load average, the total number of processes on a healthy machine usually varies just a small amount over time. Suddenly having a significantly larger or smaller number of processes could be a warning sign.

Memory: 2048M real, 1061M free, 542M swap in use, 1605M swap free

The "Memory:" line is very important. It reflects how much real and swap memory a computer has, and how much is free. "Real" memory is the amount of RAM installed in the system, a.k.a. the "physical" memory. "Swap" is virtual memory stored on the machine's disk.

Once a computer runs out of physical memory, and starts using swap space, its performance deteriorates dramatically. If you run out of swap, you'll likely crash your programs or the OS.

Individual process fields

Field	Description
PID	Process ID of process
USERNAME	Username of process
THR	Process thread PRI Priority of process
NICE	Nice value of process
SIZE	Total size of a process, including code and data, plus the stack space in kilobytes
RES	Amount of physical memory used by the process
STATE	Current CPU state of process. The states can be S for sleeping, D for uninterrupted, R for running, T for stopped/traced, and Z for zombied
TIME	The CPU time that a process has used since it started

%CPU	The CPU time that a process has used since the last update
COMMAND	The task's command name

What to Look For

- Large run queue. Large number of processes waiting in the run queue may be an indication that your system does not have sufficient CPU capacity.
- Process consuming lots of CPU. A process which is "hogging" CPU is always suspect. If this process is an oracle foreground process it's most likely running an expensive query that should be tuned. Oracle background process should not hog CPU for long periods of time.
- High load averages. Processes should not be backed up on the run queue for extended periods of time.
- Low swap space. This is an indication you are running low on memory.

Back to Contents

oswvmstat

<node_name>_vmstat_YY.MM.DD:HH24.dat

These files will contain output from the 'vmstat' command that is obtained and archive by OSWatcher Black Box at specified intervals. These files will only exist if 'vmstat' is installed on the OS and if the OSWbb user has privileges to run the utility.

The name vmstat comes from "report virtual memory statistics". The vmstat utility does a bit more than this, though. In addition to reporting virtual memory, vmstat reports certain kernel statistics about processes, disk, trap, and CPU activity.

The vmstat utility is fairly standard across UNIX platforms. Each platform will have a slightly different version of the vmstat utility. You should consult your operating system man pages for specifics. The sample provided below is for Solaris.

OSWbb runs the vmstat utility at the specified interval and stores the data in the oswvmstat subdirectory under the archive directory. The data is stored in hourly archive files. Each entry in the file contains a timestamp prefixed by *** embedded in the vmstat output. Notice there are 3 entries for each timestamp. You should always ignore the first entry as this entry is always invalid. The second and third entry will be valid but the second entry will be 1 sec later than the timestamp and the third entry will be 2 seconds later than the timestamp.

Sample vmstat file produced by OSWbb

```
***Fri Jan 28 12:50:36 EST 2005
procs memory           page                    disk       faults         cpu
r b w swap    free     re  mf   pi po fr de sr dd f0 s0    in   sy    cs   us sy  id
0 0 0 1761344 1246520 1   6    0  0  0  0  0  2  0  0  0  380  1364 900  4  1   95
0 0 0 1643920 1086776 331 1485 8  16 16 0  0  31 0  0  0  447  4966 1315 15 31  54
0 0 0 1643872 1086728 6   0    0  0  0  0  0  0  0  0  0  389  1472 932  0  0   100
```

Field Descriptions

The vmstat output is actually broken up into six sections: procs, memory, page, disk, faults and CPU. Each section is outlined in the following table.

Field	Description
PROCS	
r	Number of processes that are in a wait state and basically not doing anything but waiting to run
b	Number of processes that were in sleep mode and were interrupted since the last update

w	Number of processes that have been swapped out by mm and vm subsystems and have yet to run
MEMORY	
swap	The amount of swap space currently available free The size of the free list
PAGE	
re	page reclaims
mf	minor faults
pi	kilobytes paged in
po	kilobytes paged out
fr	kilobytes freed
de	anticipated short-term memory shortfall (Kbytes)
sr	pages scanned by clock algorithm
DISK	
Bi	Disk blocks sent to disk devices in blocks per second
FAULTS	
In	Interrupts per second, including the CPU clocks
Sy	System calls
Cs	Context switches per second within the kernel
CPU	
Us	Percentage of CPU cycles spent on user processes
Sy	Percentage of CPU cycles spent on system processes
Id	Percentage of unused CPU cycles or idle time when the CPU is basically doing nothing

What to look for

The following information should be used as a guideline and not considered hard and fast rules. The information documented below comes from Adrian Cockcroft's book, Sun Performance Tuning. Other operating systems like HP and Linux may have different thresholds.

- Large run queue. Adrian Cockcroft defines anything over 4 processes per CPU on the run queue as the threshold for CPU saturation. This is certainly a problem if this last for any long period of time.
- CPU utilization. The amount of time spent running system code should not exceed 30% especially if idle time is close to 0%.
- A combination of large run queue with no idle CPU is an indication the system has insufficient CPU capacity.
- Memory bottlenecks are determined by the scan rate (sr) . The scan rate is the pages scanned by the clock algorithm per second. If the scan rate (sr) is continuously over 200 pages per second then there is a memory shortage.
- Disk problems may be identified if the number of processes blocked exceeds the number of processes on run queue.

Back to Contents

Graphing the Output

A new utility, OSWbba has been added to OSWbb. This utility provides the ability to graph vmstat and iostat data. See the OSW Black Box Analyzer User Guide for more information. To see a sample of the OSWbba output, click here. To add database metrics use the LTOM profiler. Click here to see a sample LTOM profiler.

Sample Graph

Back to Contents

Known Issues

No issues to report.

Back to Contents

Download

Current Unix Version 4.0.0 Feburary 1, 2012

Download the latest version of OSWbb by clicking on the download link provided in Note: 301137.1. The download link no longer exists from inside this document. If you are still unable to download the file, you may request that we email you a copy: carl.davis@oracle.com

Back to Contents

Reporting Feedback

If you encounter problems running OSWbb which is not listed under the Known Issue section or would like to provide comments/feedback about OSWbb (including enhancement requests) please send email to carl.davis@oracle.com

Back to Contents

Sending Files To Support

For those users running RAC-DDT, the OSWbb archive directory will be automatically included in the RAC-DDT.tar.Z compressed archive file. For more information on RAC-DDT see <>. For users not running RAC-DDT, create a tarball of the archive directory to send in to support by executing tarupfiles.sh file in the oswbb directory.

Back to Contents