NAME

lockstat - report kernel lock and profiling statistics

SYNOPSIS

lockstat [**-ACEHIV**] [**-e** *event-list*] [**-i** *rate*] [**-b** | **-t** | **-h** | **-s** *depth*] [**-n** *num-records*] [**-l** *lock* [,*size*]] [**-d** *duration*] [**-f** *function* [,*size*]] [**-T**] [**-kgwWRpP**] [**-D** *count*] [**-o** **-filename**] [**-x** *opt* [=*val*]] *command* [[*args*]]

DESCRIPTION

The **lockstat** utility gathers and displays kernel locking and profiling statistics. **lockstat** allows you to specify which events to watch (for example, spin on adaptive mutex, block on read access to rwlock due to waiting writers, and so forth), how much data to gather for each event, and how to display the data. By default, **lockstat** monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

lockstat gathers data until the specified command completes. For example, to gather statistics for a fixed-time interval, use sleep(1) as the command, as follows:

```
# lockstat sleep 5
```

When the **-I** option is specified, **lockstat** establishes a per-processor high-level periodic interrupt source to gather profiling data. The interrupt handler simply generates a **lockstat** event whose caller is the interrupted PC (program counter). The profiling event is just like any other **lockstat** event, so all of the normal **lockstat** options are applicable.

lockstat relies on DTrace to modify the running kernel’s text to intercept events of interest. This imposes a small but measurable overhead on all system activity, so access to **lockstat** is restricted to super-user by default.

OPTIONS

The following options are supported:

-V Print the D program used to gather the requested data.

Event Selection

If no event selection options are specified, the default is **-C**.

-A Watch all lock events. **-A** is equivalent to **-CH**.

-C Watch contention events.

-E Watch error events.

-e *event-list*

Only watch the specified events. *event-list* is a comma-separated list of events or ranges of events such as 1,4-7,35. Run **lockstat** with no arguments to get a brief description of all events.

-H Watch hold events.

-I Watch profiling interrupt events.

-i *rate* Interrupt rate (per second) for **-I**. The default is 97 Hz, so that profiling doesn't run in lockstep with the clock interrupt (which runs at 100 Hz).

Data Gathering

-x *arg* [=*val*]

Enable or modify a dtrace(1) runtime option or D compiler option. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign.

Data Gathering (Mutually Exclusive)

-b Basic statistics: lock, caller, number of events.

-h Histogram: timing plus time-distribution histograms.

-s *depth*

Stack trace: histogram plus stack traces up to *depth* frames deep.

-t Timing: Basic plus timing for all events (default).

Data Filtering

-d *duration*

Only watch events longer than *duration*.

-f *func*[,*size*]

Only watch events generated by *func*, which can be specified as a symbolic name or hex address. *size* defaults to the ELF symbol size if available, or 1 if not.

-l *lock*[,*size*]

Only watch *lock*, which can be specified as a symbolic name or hex address. *size* defaults to the ELF symbol size or 1 if the symbol size is not available.

-n *num-records*
Maximum number of data records.

-T Trace (rather than sample) events. This is off by default.

Data Reporting

-D *count*
Only display the top *count* events of each type.

-g Show total events generated by function. For example, if **foo**() calls **bar**() in a loop, the work done by **bar**() counts as work generated by **foo**() (along with any work done by **foo**() itself). The **-g** option works by counting the total number of stack frames in which each function appears. This implies two things: (1) the data reported by **-g** can be misleading if the stack traces are not deep enough, and (2) functions that are called recursively might show greater than 100% activity. In light of issue (1), the default data gathering mode when using **-g** is **-s -50**.

-k Coalesce PCs within functions.

-o *filename*
Direct output to *filename*.

-P Sort data by (*count * time*) product.

-p Parsable output format.

-R Display rates (events per second) rather than counts.

-W Whichever: distinguish events only by caller, not by lock.

-w Wherever: distinguish events only by lock, not by caller.

DISPLAY FORMATS

The following headers appear over various columns of data.

Count or ops/s
Number of times this event occurred, or the rate (times per second) if **-R** was specified.

indv Percentage of all events represented by this individual event.

genr Percentage of all events generated by this function.

cuml Cumulative percentage; a running total of the individuals.

rcnt Average reference count. This will always be 1 for exclusive locks (mutexes, spin locks, rwlocks held as writer) but can be greater than 1 for shared locks (rwlocks held as reader).

nsec Average duration of the events in nanoseconds, as appropriate for the event. For the profiling event, duration means interrupt latency.

Lock Address of the lock; displayed symbolically if possible.

CPU+Pri_Class
CPU plus the priority class of the interrupted thread. For example, if CPU 4 is interrupted while running a timeshare thread, this will be reported as ‘cpu[4]+TShar’.

Caller Address of the caller; displayed symbolically if possible.

EXAMPLES

Example 1 Measuring Kernel Lock Contention

```
# lockstat sleep 5

Adaptive mutex spin: 41411 events in 5.011 seconds (8263 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
13750  33%  33% 0.00       72 vm_page_queue_free_mtx vm_page_free_toq+0x12e
13648  33%  66% 0.00       66 vm_page_queue_free_mtx vm_page_alloc+0x138
 4023  10%  76% 0.00       51 vm_dom+0x80            vm_page_dequeue+0x68
 2672   6%  82% 0.00      186 vm_dom+0x80            vm_page_enqueue+0x63
  618   1%  84% 0.00       31 0xfffff8000cd83a88     qsyncvp+0x37
  506   1%  85% 0.00      164 0xfffff8000cb3f098     vputx+0x5a
  477   1%  86% 0.00       69 0xfffff8000c7eb180     uma_dbg_getslab+0x5b
  288   1%  87% 0.00       77 0xfffff8000cd8b000     vn_finished_write+0x29
  263   1%  88% 0.00      103 0xfffff8000cbad448     vinactive+0xdc
  259   1%  88% 0.00       53 0xfffff8000cd8b000     vfs_ref+0x24
  237   1%  89% 0.00       20 0xfffff8000cbad448     vfs_hash_get+0xcc
  233   1%  89% 0.00       22 0xfffff8000bfd9480     uma_dbg_getslab+0x5b
  223   1%  90% 0.00       20 0xfffff8000cb3f098     cache_lookup+0x561
  193   0%  90% 0.00       16 0xfffff8000cb40ba8     vref+0x27
  175   0%  91% 0.00       34 0xfffff8000cbad448     vputx+0x5a
  169   0%  91% 0.00       51 0xfffff8000cd8b000     vfs_unbusy+0x27
  164   0%  92% 0.00       31 0xfffff8000cb40ba8     vputx+0x5a
[...]

Adaptive mutex block: 10 events in 5.011 seconds (2 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
    3  30%  30% 0.00    17592 vm_page_queue_free_mtx vm_page_alloc+0x138
    2  20%  50% 0.00    20528 vm_dom+0x80            vm_page_enqueue+0x63
    2  20%  70% 0.00    55502 0xfffff8000cb40ba8     vputx+0x5a
    1  10%  80% 0.00    12007 vm_page_queue_free_mtx vm_page_free_toq+0x12e
    1  10%  90% 0.00     9125 0xfffff8000cbad448     vfs_hash_get+0xcc
    1  10% 100% 0.00     7864 0xfffff8000cd83a88     qsyncvp+0x37
-------------------------------------------------------------------------------
[...]
```

Example 2 Measuring Hold Times

```
# lockstat -H -D 10 sleep 1

Adaptive mutex hold: 109589 events in 1.039 seconds (105526 events/sec)

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
 8998   8%   8% 0.00      617 0xfffff8000c7eb180     uma_dbg_getslab+0xd4
 5901   5%  14% 0.00      917 vm_page_queue_free_mtx vm_object_terminate+0x16a
 5040   5%  18% 0.00      902 vm_dom+0x80            vm_page_free_toq+0x88
 4884   4%  23% 0.00     1056 vm_page_queue_free_mtx vm_page_alloc+0x44e
 4664   4%  27% 0.00      759 vm_dom+0x80            vm_fault_hold+0x1a13
 4011   4%  31% 0.00      888 vm_dom                 vm_page_advise+0x11b
 4010   4%  34% 0.00      957 vm_dom+0x80            _vm_page_deactivate+0x5c
 3743   3%  38% 0.00      582 0xfffff8000cf04838     pmap_is_prefaultable+0x158
 2254   2%  40% 0.00      952 vm_dom                 vm_page_free_toq+0x88
 1639   1%  41% 0.00      591 0xfffff800d60065b8     trap_pfault+0x1f7
-------------------------------------------------------------------------------
[...]

R/W writer hold: 64314 events in 1.039 seconds (61929 events/sec)
```

```
Count indv cuml rcnt     nsec Lock                 Caller
-------------------------------------------------------------------------------
 7421  12%  12% 0.00     2994 pvh_global_lock        pmap_page_is_mapped+0xb6
 4668   7%  19% 0.00     3313 pvh_global_lock        pmap_enter+0x9ae
 1639   3%  21% 0.00      733 0xfffff80168d10200     vm_object_deallocate+0x683
 1639   3%  24% 0.00     3061 0xfffff80168d10200     unlock_and_deallocate+0x2b
 1639   3%  26% 0.00     2966 0xfffff80168d10200     vm_fault_hold+0x16ee
 1567   2%  29% 0.00      733 0xfffff80168d10200     vm_fault_hold+0x19bc
  821   1%  30% 0.00      786 0xfffff801eb0cc000     vm_object_madvise+0x32d
  649   1%  31% 0.00     4918 0xfffff80191105300     vm_fault_hold+0x16ee
  648   1%  32% 0.00     8112 0xfffff80191105300     unlock_and_deallocate+0x2b
  647   1%  33% 0.00     1261 0xfffff80191105300     vm_object_deallocate+0x683
-------------------------------------------------------------------------------
```

Example 3 Measuring Hold Times for Stack Traces Containing a Specific Function

```
# lockstat -H -f tcp_input -s 50 -D 10 sleep 1

Adaptive mutex hold: 68 events in 1.026 seconds (66 events/sec)

-------------------------------------------------------------------------------
Count indv cuml rcnt     nsec Lock                 Caller
   32  47%  47% 0.00     1631 0xfffff800686f50d8     tcp_do_segment+0x284b

      nsec ------ Time Distribution ------ count     Stack
      1024 |@@@@@@@@@@                     11        tcp_input+0xf54
      2048 |@@@@@@@@@@@@@                  14        ip_input+0xc8
      4096 |@@@@@                          6         swi_net+0x192
      8192 |                               1         intr_event_execute_handlers+0x93
                                                     ithread_loop+0xa6
                                                     fork_exit+0x84
                                                     0xffffffff808cf9ee
-------------------------------------------------------------------------------
Count indv cuml rcnt     nsec Lock                 Caller
   29  43%  90% 0.00     4851 0xfffff800686f50d8     sowakeup+0xf8

      nsec ------ Time Distribution ------ count     Stack
      4096 |@@@@@@@@@@@@@@@                15        tcp_do_segment+0x2423
      8192 |@@@@@@@@@@@@                   12        tcp_input+0xf54
     16384 |@@                             2         ip_input+0xc8
```

```
                                    swi_net+0x192
                                    intr_event_execute_handlers+0x93
                                    ithread_loop+0xa6
                                    fork_exit+0x84
                                    0xffffffff808cf9ee
    -------------------------------------------------------------------------------
    [...]
```

SEE ALSO

dtrace(1), ksyms(4), locking(9)

HISTORY

The **lockstat** utility first appeared in FreeBSD 7.1.

NOTES

Tail-call elimination can affect call sites. For example, if **foo**()+0x50 calls **bar**() and the last thing **bar**() does is call **mtx_unlock**(), the compiler can arrange for **bar**() to branch to **mtx_unlock**() with a return address of **foo**()+0x58. Thus, the **mtx_unlock**() in **bar**() will appear as though it occurred at **foo**()+0x58.

The PC in the stack frame in which an interrupt occurs can be bogus because, between function calls, the compiler is free to use the return address register for local storage.

When using the **-I** and **-s** options together, the interrupted PC will usually not appear anywhere in the stack since the interrupt handler is entered asynchronously, not by a function call from that PC.