Lock Statistics

What

As the name suggests, it provides statistics on locks.

Why

Because things like lock contention can severely impact performance.

How

Lockdep already has hooks in the lock functions and maps lock instances to lock classes. We build on that (see Documentation/locking/lockdep-design.rst). The graph below shows the relation between the lock functions and the various hooks therein:

```
         __acquire
             |
            lock _____
             |        \
             |    __contended
             |         |
             |       <wait>
             | _______/
             |/
             |
       __acquired
             |
             .
           <hold>
             .
             |
       __release
             |
           unlock

lock, unlock  - the regular lock functions
__*           - the hooks
<>            - states
```

With these hooks we provide the following statistics:

con-bounces
- number of lock contention that involved x-cpu data

contentions
- number of lock acquisitions that had to wait

wait time
  min
    - shortest (non-0) time we ever had to wait for a lock
  max
    - longest time we ever had to wait for a lock
  total
    - total time we spend waiting on this lock
  avg
    - average time spent waiting on this lock

acq-bounces
- number of lock acquisitions that involved x-cpu data

acquisitions
- number of times we took the lock

hold time
  min
    - shortest (non-0) time we ever held the lock
  max
    - longest time we ever held the lock
  total
    - total time this lock was held
  avg
    - average time this lock was held

These numbers are gathered per lock class, per read/write state (when applicable).

It also tracks 4 contention points per class. A contention point is a call site that had to wait on lock acquisition.

Configuration

Lock statistics are enabled via CONFIG_LOCK_STAT.

Usage

Enable collection of statistics:

```
# echo 1 >/proc/sys/kernel/lock_stat
```

Disable collection of statistics:

```
# echo 0 >/proc/sys/kernel/lock_stat
```

Look at the current lock statistics:

```
( line numbers not part of actual output, done for clarity in the explanation
  below )

# less /proc/lock_stat

01 lock_stat version 0.4
02-----------------------------------------------------------------------------------------------------------------------------------------------
03                              class name    con-bounces    contentions   waittime-min   waittime-max waittime-total   waittime-avg    a
04-----------------------------------------------------------------------------------------------------------------------------------------------
05
06                          &mm->mmap_sem-W:            46             84           0.26         939.10       16371.53         194.90
07                          &mm->mmap_sem-R:            37            100           1.31      299502.61      325629.52        3256.30
08                          ---------------
09                            &mm->mmap_sem             1          [<ffffffff811502a7>] khugepaged_scan_mm_slot+0x57/0x280
10                            &mm->mmap_sem            96          [<ffffffff815351c4>] __do_page_fault+0x1d4/0x510
11                            &mm->mmap_sem            34          [<ffffffff81113d77>] vm_mmap_pgoff+0x87/0xd0
12                            &mm->mmap_sem            17          [<ffffffff81127e71>] vm_munmap+0x41/0x80
13                          ---------------
14                            &mm->mmap_sem             1          [<ffffffff81046fda>] dup_mmap+0x2a/0x3f0
15                            &mm->mmap_sem            60          [<ffffffff81129e29>] SyS_mprotect+0xe9/0x250
16                            &mm->mmap_sem            41          [<ffffffff815351c4>] __do_page_fault+0x1d4/0x510
17                            &mm->mmap_sem            68          [<ffffffff81113d77>] vm_mmap_pgoff+0x87/0xd0
18
19.............................................................................................................................................
20
21                          unix_table_lock:           110            112           0.21          49.24         163.91           1.46
```

```
22                          ---------------
23                          unix_table_lock            45          [<ffffffff8150ad8e>] unix_create1+0x16e/0x1b0
24                          unix_table_lock            47          [<ffffffff8150b111>] unix_release_sock+0x31/0x250
25                          unix_table_lock            15          [<ffffffff8150ca37>] unix_find_other+0x117/0x230
26                          unix_table_lock             5          [<ffffffff8150a09f>] unix_autobind+0x11f/0x1b0
27                          ---------------
28                          unix_table_lock            39          [<ffffffff8150b111>] unix_release_sock+0x31/0x250
29                          unix_table_lock            49          [<ffffffff8150ad8e>] unix_create1+0x16e/0x1b0
30                          unix_table_lock            20          [<ffffffff8150ca37>] unix_find_other+0x117/0x230
31                          unix_table_lock             4          [<ffffffff8150a09f>] unix_autobind+0x11f/0x1b0
```

This excerpt shows the first two lock class statistics. Line 01 shows the output version - each time the format changes this will be updated. Line 02-04 show the header with column descriptions. Lines 05-18 and 20-31 show the actual statistics. These statistics come in two parts; the actual stats separated by a short separator (line 08, 13) from the contention points.

Lines 09-12 show the first 4 recorded contention points (the code which tries to get the lock) and lines 14-17 show the first 4 recorded contended points (the lock holder). It is possible that the max con-bounces point is missing in the statistics.

The first lock (05-18) is a read/write lock, and shows two lines above the short separator. The contention points don't match the column descriptors, they have two: contentions and [<IP>] symbol. The second set of contention points are the points we're contending with.

The integer part of the time values is in us.

Dealing with nested locks, subclasses may appear:

```
32.........................................................................................................................
33
34                              &rq->lock:       13128          13128           0.43         190.53      103881.26           7.91
35                              ---------
36                              &rq->lock          645          [<ffffffff8103bfc4>] task_rq_lock+0x43/0x75
37                              &rq->lock          297          [<ffffffff8104ba65>] try_to_wake_up+0x127/0x25a
38                              &rq->lock          360          [<ffffffff8103c4c5>] select_task_rq_fair+0x1f0/0x74a
39                              &rq->lock          428          [<ffffffff81045f98>] scheduler_tick+0x46/0x1fb
40                              ---------
41                              &rq->lock           77          [<ffffffff8103bfc4>] task_rq_lock+0x43/0x75
42                              &rq->lock          174          [<ffffffff8104ba65>] try_to_wake_up+0x127/0x25a
43                              &rq->lock         4715          [<ffffffff8103ed4b>] double_rq_lock+0x42/0x54
44                              &rq->lock          893          [<ffffffff81340524>] schedule+0x157/0x7b8
45
46.........................................................................................................................
47
48                            &rq->lock/1:        1526          11488           0.33         388.73      136294.31          11.86
49                            -----------
50                            &rq->lock/1        11526          [<ffffffff8103ed58>] double_rq_lock+0x4f/0x54
51                            -----------
52                            &rq->lock/1         5645          [<ffffffff8103ed4b>] double_rq_lock+0x42/0x54
53                            &rq->lock/1         1224          [<ffffffff81340524>] schedule+0x157/0x7b8
54                            &rq->lock/1         4336          [<ffffffff8103ed58>] double_rq_lock+0x4f/0x54
55                            &rq->lock/1          181          [<ffffffff8104ba65>] try_to_wake_up+0x127/0x25a
```

Line 48 shows statistics for the second subclass (/1) of &rq->lock class (subclass starts from 0), since in this case, as line 50 suggests, double_rq_lock actually acquires a nested lock of two spinlocks.

View the top contending locks:

```
# grep : /proc/lock_stat | head
                      clockevents_lock:       2926159        2947636           0.15       46882.81  1784540466.34         605.41
                    tick_broadcast_lock:        346460         346717           0.18        2257.43    39364622.71         113.54
                  &mapping->i_mmap_mutex:        203896         203899           3.36      645530.05 31767507988.39      155800.21
                                &rq->lock:        135014         136909           0.18         606.09      842160.68           6.15
              &(&zone->lru_lock)->rlock:         93000          94934           0.16          59.18      188253.78           1.98
                         tasklist_lock-W:         40667          41130           0.23        1189.42      428980.51          10.43
                         tasklist_lock-R:         21298          21305           0.20        1310.05      215511.12          10.12
                               rcu_node_1:         47656          49022           0.16         635.41      193616.41           3.95
     &(&dentry->d_lockref.lock)->rlock:         39791          40179           0.15        1302.08       88851.96           2.21
                               rcu_node_0:         29203          30064           0.16         786.55     1555573.00          51.74
```

Clear the statistics:

```
# echo 0 > /proc/lock_stat
```