Version 14 of schedstats includes support for sched_domains, which hit the mainline kernel in 2.6.20 although it is identical to the stats from version 12 which was in the kernel from 2.6.13-2.6.19 (version 13 never saw a kernel release). Some counters make more sense to be per-runqueue; other to be per-domain. Note that domains (and their associated information) will only be pertinent and available on machines utilizing CONFIG_SMP.

In version 14 of schedstat, there is at least one level of domain statistics for each cpu listed, and there may well be more than one domain. Domains have no particular names in this implementation, but the highest numbered one typically arbitrates balancing across all the cpus on the machine, while domain0 is the most tightly focused domain, sometimes balancing only between pairs of cpus. At this time, there are no architectures which need more than three domain levels. The first field in the domain stats is a bit map indicating which cpus are affected by that domain.

These fields are counters, and only increment. Programs which make use of these will need to start with a baseline observation and then calculate the change in the counters at each subsequent observation. A perl script which does this for many of the fields is available at

http://eaglet.rain.com/rick/linux/schedstat/

Note that any such script will necessarily be version-specific, as the main reason to change versions is changes in the output format. For those wishing to write their own scripts, the fields are described here.

CPU statistics
---

cpu<N> 1 2 3 4 5 6 7 8 9 10 11 12

NOTE: In the sched_yield() statistics, the active queue is considered empty if it has only one process in it, since obviously the process calling sched_yield() is that process.

First four fields are sched_yield() statistics:
1) # of times both the active and the expired queue were empty
2) # of times just the active queue was empty
3) # of times just the expired queue was empty
4) # of times sched_yield() was called

Next three are schedule() statistics:
5) # of times we switched to the expired queue and reused it
6) # of times schedule() was called
7) # of times schedule() left the processor idle

Next two are try_to_wake_up() statistics:
8) # of times try_to_wake_up() was called
9) # of times try_to_wake_up() was called to wake up the local cpu

Next three are statistics describing scheduling latency:
10) sum of all time spent running by tasks on this processor (in jiffies)
11) sum of all time spent waiting to run by tasks on this processor (in jiffies)
12) # of timeslices run on this cpu

Domain statistics

One of these is produced per domain for each cpu described. (Note that if CONFIG_SMP is not defined, *no* domains are utilized and these lines will not appear in the output.)

domain<N> <cpumask> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

The first field is a bit mask indicating what cpus this domain operates over.

The next 24 are a variety of load_balance() statistics in grouped into types of idleness (idle, busy, and newly idle):

1) # of times in this domain load_balance() was called when the cpu was idle
2) # of times in this domain load_balance() checked but found the load did not require balancing when the cpu was idle
3) # of times in this domain load_balance() tried to move one or more tasks and failed, when the cpu was idle
4) sum of imbalances discovered (if any) with each call to load_balance() in this domain when the cpu was idle
5) # of times in this domain pull_task() was called when the cpu was idle
6) # of times in this domain pull_task() was called even though the target task was cache-hot when idle
7) # of times in this domain load_balance() was called but did not find a busier queue while the cpu was idle
8) # of times in this domain a busier queue was found while the cpu was idle but no busier group was found

9) # of times in this domain load_balance() was called when the cpu was busy
10) # of times in this domain load_balance() checked but found the load did not require balancing when busy
11) # of times in this domain load_balance() tried to move one or more tasks and failed, when the cpu was busy
12) sum of imbalances discovered (if any) with each call to load_balance() in this domain when the cpu was busy
13) # of times in this domain pull_task() was called when busy
14) # of times in this domain pull_task() was called even though the target task was cache-hot when busy
15) # of times in this domain load_balance() was called but did not find a busier queue while the cpu was busy
16) # of times in this domain a busier queue was found while the cpu was busy but no busier group was found

17) # of times in this domain load_balance() was called when the cpu was just becoming idle
18) # of times in this domain load_balance() checked but found the load did not require balancing when the cpu was just becoming idle
19) # of times in this domain load_balance() tried to move one or more tasks and failed, when the cpu was just becoming idle
20) sum of imbalances discovered (if any) with each call to

load_balance() in this domain when the cpu was just becoming idle
21) # of times in this domain pull_task() was called when newly idle
22) # of times in this domain pull_task() was called even though the target task was cache-hot when just becoming idle
23) # of times in this domain load_balance() was called but did not find a busier queue while the cpu was just becoming idle
24) # of times in this domain a busier queue was found while the cpu was just becoming idle but no busier group was found

Next three are active_load_balance() statistics:
25) # of times active_load_balance() was called
26) # of times active_load_balance() tried to move a task and failed
27) # of times active_load_balance() successfully moved a task

Next three are sched_balance_exec() statistics:
28) sbe_cnt is not used
29) sbe_balanced is not used
30) sbe_pushed is not used

Next three are sched_balance_fork() statistics:
31) sbf_cnt is not used
32) sbf_balanced is not used
33) sbf_pushed is not used

Next three are try_to_wake_up() statistics:
34) # of times in this domain try_to_wake_up() awoke a task that last ran on a different cpu in this domain
35) # of times in this domain try_to_wake_up() moved a task to the waking cpu because it was cache-cold on its own cpu anyway
36) # of times in this domain try_to_wake_up() started passive balancing

/proc/<pid>/schedstat
---

schedstats also adds a new /proc/<pid>/schedstat file to include some of the same information on a per-process level. There are three fields in this file correlating for that process to:
1) time spent on the cpu
2) time spent waiting on a runqueue
3) # of timeslices run on this cpu

A program could be easily written to make use of these extra fields to report on how well a particular process or set of processes is faring under the scheduler's policies. A simple version of such a program is available at
http://eaglet.rain.com/rick/linux/schedstat/v12/latency.c