Scheduler debugfs

Booting a kernel with CONFIG_SCHED_DEBUG=y will give access to scheduler specific debug files under /sys/kernel/debug/sched. Some of those files are described below.

numa_balancing

numa_balancing directory is used to hold files to control NUMA balancing feature. If the system overhead from the feature is too high then the rate the kernel samples for NUMA hinting faults may be controlled by the *scan_period_min_ms, scan_delay_ms, scan_period_max_ms, scan_size_mb* files.

scan_period_min_ms, scan_delay_ms, scan_period_max_ms, scan_size_mb

Automatic NUMA balancing scans tasks address space and unmaps pages to detect if pages are properly placed or if the data should be migrated to a memory node local to where the task is running. Every "scan delay" the task scans the next "scan size" number of pages in its address space. When the end of the address space is reached the scanner restarts from the beginning.

In combination, the "scan delay" and "scan size" determine the scan rate. When "scan delay" decreases, the scan rate increases. The scan delay and hence the scan rate of every task is adaptive and depends on historical behaviour. If pages are properly placed then the scan delay increases, otherwise the scan delay decreases. The "scan size" is not adaptive but the higher the "scan size", the higher the scan rate.

Higher scan rates incur higher system overhead as page faults must be trapped and potentially data must be migrated. However, the higher the scan rate, the more quickly a tasks memory is migrated to a local node if the workload pattern changes and minimises performance impact due to remote memory accesses. These files control the thresholds for scan delays and the number of pages scanned.

`scan_period_min_ms` is the minimum time in milliseconds to scan a tasks virtual memory. It effectively controls the maximum scanning rate for each task.

`scan_delay_ms` is the starting "scan delay" used for a task when it initially forks.

`scan_period_max_ms` is the maximum time in milliseconds to scan a tasks virtual memory. It effectively controls the minimum scanning rate for each task.

`scan_size_mb` is how many megabytes worth of pages are scanned for a given scan.