The interactivity score is compared to the interactivity threshold, which is the cutoff point for considering a thread interactive. The interactivity threshold is modified by the process ***nice*** value. Positive nice values make it more challenging for a thread to be considered interactive, while negative values make it easier. Thus, the nice value gives the user some control over the primary mechanism of reducing thread-scheduling latency.

A thread is considered to be interactive if the ratio of its voluntary sleep time versus its run time is below a certain threshold. The interactivity threshold is defined in the ULE code and is not configurable. ULE uses two equations to compute the interactivity score of a thread. For threads whose sleep time exceeds their run time, Eq 4.1 is used:

$$interactivity\ score = \frac{scaling\ \ factor}{sleep\ /\ run} \qquad \text{(Eq. 4.1)}$$

When a thread's run time exceeds its sleep time, Eq. 4.2 is used instead:

$$interactivity\ score = \frac{scaling\ \ factor}{run\ /\ sleep} + scaling\ \ factor \qquad \text{(Eq. 4.2)}$$

The scaling factor is the maximum interactivity score divided by two. Threads that score below the interactivity threshold are considered to be interactive; all others are noninteractive. The *sched_interact_update*() routine is called at several points in a threads existence—for example, when the thread is awakened by a *wakeup*() call—to update the thread's run time and sleep time. The sleep- and run-time values are only allowed to grow to a certain limit. When the sum of the run time and sleep time pass the limit, they are reduced to bring them back into range. An interactive thread whose sleep history was not remembered at all would not remain interactive, resulting in a poor user experience. Remembering an interactive thread's sleep time for too long would allow the thread to get more than its fair share of the CPU. The amount of history that is kept and the interactivity threshold are the two values that most strongly influence a user's interactive experience on the system.

Priorities are assigned according to the thread's interactivity status. Interactive threads have a priority that is derived from the interactivity score and are placed in a priority band above batch threads. They are scheduled like real-time round-robin threads. Batch threads have their priorities determined by the estimated CPU utilization modified according to their process nice value. In both cases, the available priority range is equally divided among possible interactive scores or percent-cpu calculations, both of which are values between 0 and 100. Since there are fewer than 100 priorities available for each class, some values share priorities. Both computations roughly assign priorities according to a history of CPU utilization but with different longevities and scaling factors.

The CPU utilization estimator accumulates run time as a thread runs and decays it as a thread sleeps. The utilization estimator provides the percent-cpu values displayed in **top** and **ps**. ULE delays the decay until a thread wakes to avoid periodically scanning every thread in the system. Since this delay leaves