CPU Scheduling in FreeBSD

Peter McKay
FreeBSD

- FreeBSD - UNIX-like OS that is descended from version 4.3 of
- BSD UNIX (forked c. 1993), which is itself descended from
History of the FreeBSD scheduler

In the beginning, FreeBSD inherited the 4BSD scheduler (itself a modification of the traditional UNIX scheduler)

The FreeBSD community eventually formulated a modified version with several enhancements, including support for symmetric multiprocessing.
A complete overhaul

The FreeBSD scheduler performed better than its ancestor, but had a few issues, which ultimately necessitated the creation of an entirely new scheduler.

FreeBSD version 5 saw the release of ULE, but it did not become the default scheduler until it reached maturity around FreeBSD v.7.1.
The ULE Scheduler

Found by typing `man sched_ule`

Cute.

- Support for SMT and SMP
- Support for CPU affinity
- O(1) for process count
- Improved interactive performance under heavy load
Great Big Fancy List of Components

The GBFLC enumerates the features of the scheduler, which will be dealt with in more detail forthwith.

- queues of queues
- a pair of load-balancing algorithms
- interactivity detection
- CPU usage estimation
- slice calculator (variable time slices?)
- priority calculator
Queue Queue

Each CPU has a KSeq:
three arrays of run queues indexed by priority

- The Current queue receives interactive, real-time, and interrupt threads.
- The Next queue receives everything else except for idle threads.
- When the Current queue is empty, the two queues swap.
- The third queue holds idle threads, and is only used when there are no other runnable tasks.
Interactivity Scoring

The interactivity of a thread is determined by monitoring its voluntary sleep time and run time. The output for Fig. 1’s algorithm is bounded by configurable values.

\[ m = \frac{\text{(Maximum Interactive Score)}}{2} \]

\[ \text{if} \ (\text{sleep} > \text{run}) \quad \text{score} = \frac{m}{\frac{\text{sleep}}{\text{run}}} \]

\[ \text{else} \quad \text{score} = \frac{m}{\frac{\text{run}}{\text{sleep}}} + m \]

*Figure 1: Interactivity scoring algorithm*
Priority Calculator

Priority is the product of Interactivity as well as the Nice factor.

Generally, interactive tasks run faster than non-interactive tasks on the same queue.

But a non-interactive thread may still run ahead of a sufficiently expensive interactive one.
Nice Impact / Slice Calculator

Moving window of nice values that are allowed slices. Determined by tracking least nice, and dragging a window behind it.

Generally only applied to time-sharing processes (not real-time, interactive, interrupts, idle threads)
CPU Usage Estimation

Keeps track of the number of clock ticks in a sliding window around a thread’s execution time.

Since it’s an event-driven scheduler, an event had to be added to the scheduler that updates the window and tick count whenever usage is read.
SMP

Symmetric MultiProcessor system.

- Idle CPUs can steal threads from busier CPUs
- Busy CPUs periodically (twice per second) check the load and may distribute some of their load to other CPUs.
SMT

Simultaneous Multi-Threading.

Pipelining of multiple diverse threads on a single physical unit. Usually accomplished by presenting a single SMT core as multiple logical cores to the scheduler.

Logical cores are treated as less powerful than physical cores, by assigning multiple logical cores to a single scheduling object
So, what does this buy us?

A ULE system, on optimistic cases, completes certain tasks four times faster than an identical system running the 4BSD scheduler. On more realistic input, an increase in throughput of up to 30% has been observed (and this is in the early days).