Scheduling II

- Multiprocessor scheduling issues
- Real-time scheduling
- Linux scheduling
- Linux scheduler architecture
How to allocate processes to CPUs?
Symmetric multiprocessing (SMP)

- Multiple CPUs
- Same access time to main memory
- Private cache
Global queue of processes

- One ready queue shared across all CPUs

- Advantages
  - Good CPU utilization
  - Fair to all processes

- Disadvantages
  - Not scalable (contention for global queue lock)
  - Poor cache locality

- Linux 2.4 uses global queue
Per-CPU queue of processes

- Static partition of processes to CPUs

- Advantages
  - Easy to implement
  - Scalable (no contention on ready queue)
  - Better cache locality

- Disadvantages
  - Load-imbalance (some CPUs have more processes)
    - Unfair to processes and lower CPU utilization
Modern OSes take hybrid approaches

- Use both global and per-CPU queues
- Migrate processes across per-CPU queues

Processor Affinity
  - Add process to a CPU’s queue if recently run on the CPU
    - Cache state may still present
Real-time scheduling

- Real-time processes have timing constraints
  - Expressed as deadlines or rate requirements
  - Ex) gaming, video/music player, autopilot

- **Hard real-time** systems – required to complete a critical task within a guaranteed amount of time

- **Soft real-time** computing – requires that critical processes receive priority over others

- Linux supports soft real-time
Linux: multi-level queue with priorities

- **Soft real-time scheduling policies**
  - `SCHED_FIFO` (FCFS)
  - `SCHED_RR` (round robin)
  - Priority over normal tasks
  - 100 static priority levels (1..99)

- **Normal scheduling policies**
  - `SCHED_NORMAL`: standard
    - `SCHED_OTHER` in POSIX
  - `SCHED_BATCH`: CPU bound
  - `SCHED_IDLE`: lower priority
  - Static priority is 0
    - 40 dynamic priority
    - “Nice” values

- `sched_setscheduler()`, `nice()`
- See “man 7 sched” for detailed overview
Linux scheduler history

- **O(N) scheduler up to 2.4**
  - Simple: global run queue
  - Poor performance on multiprocessor and large N

- **O(1) scheduler in 2.5 & 2.6**
  - Good performance: per-CPU run queue
  - Complex and error prone logic to boost interactivity
  - No fairness guarantee

- **Completely Fair Scheduler (CFS) in 2.6 and later**
  - Currently default scheduler for SCHED_NORMAL
  - Processes get fair share of CPU
  - Naturally boosts interactivity

- **BFS and MuQSS**
  - Linux scheduler for hippies
  - Available as kernel patches on the street
Ideal fair scheduling

- Infinitesimally small time slice
- $n$ processes: each runs uniformly at $1/n^{\text{th}}$ rate

**1 Process**

**3 Processes**

1/3rd progress

Various approximations of the ideal

- Lottery scheduling
- Stride scheduling
- Linux CFS
Completely Fair Scheduler (CFS)

- **Approximate fair scheduling**
  - Run each process once per schedule latency period
    - `sysctl_sched_latency`
  - Time slice for process Pi: $T * \frac{W_i}{\text{Sum of all } W_i}$
    - `sched_slice()`
- **Too many processes?**
  - Lower bound on smallest time slice
  - Schedule latency = lower bound * number of procs
- **Introduced in Linux 2.6.23**
Picking the next process

- Pick proc with weighted minimum runtime so far
  - Virtual runtime: `task->vruntime += executed time / Wi`

- Example
  - P1: 1 ms burst per 10 ms (schedule latency)
  - P2 and P3 are CPU-bound
  - All processes have the same weight (1)
Finding proc with minimum runtime fast

- **Red-black tree**
  - Balanced binary search tree
  - Ordered by vruntime as key
  - $O(\lg N)$ insertion, deletion, update, $O(1)$: find min

- Tasks move from left of tree to the right
- **min_vruntime** caches smallest value
- Update vruntime and **min_vruntime**
  - When task is added or removed
  - On every timer tick, context switch
Converting nice level to weight

- Table of nice level to weight
  - static const int prio_to_weight[40] (kernel/sched/sched.h)

- Nice level changes by 1 ➔ 10% weight

- Pre-computed to avoid
  - Floating point operations
  - Runtime overhead
Fsck all that...

Enter BFS

The scheduler that shall not be named

(now replaced by MuQSS, sadly...)