Scheduling in Unix

• Pure time-sharing operating system
• Uses a multilevel adaptive scheduling policy
  – In Unix 4.3 BSD, priorities are in the range 0 to 127
    • Processes in user mode have priorities between 50 and 127
    • Processes in kernel mode have priorities between 0 and 49
  – Process priority = base priority for user processes
    \[ + f (\text{CPU time used recently}) + \text{nice value} \]
    • A process can decrease its priority by setting a positive nice value
• For fair share
  – Add the term \( f \) (CPU time used by processes in group)

Scheduling in Unix (continued)

• PCB contains \( CPU \ usage \) field, initialized to 0
• Clock raises an interrupt 60 times a second
  – CPU usage field of running process is incremented by 1 when an interrupt occurs
• Scheduler recomputes process priorities every second
  – For each process, value in CPU usage field is divided by 2, stored back, and used as the value of \( f \)
  – The effect of CPU time used by a process decays over time \( \rightarrow \) avoid starvation
Scheduling in Windows

• Scheduling is priority-driven and preemptive
  – Real-time threads are given higher priorities (16-31), other threads have priorities 1-15
  – Within a priority level, RR scheduling with time-slicing
  – Priorities of non-real-time threads are dynamically varied
    • Priority depends on: base priority of process, base priority of thread, and a dynamic component to favor interactive threads

Scheduling in Windows (continued)

• Dynamic component of priority is varied as follows
  – Lowered when a thread uses up its complete time slice \(\Rightarrow\) limit CPU consumption of CPU-bound threads
  – Increased when a blocked thread is activated \(\Rightarrow\) give good response time to interactive threads

• To prevent starvation
  – If a ready thread has not received CPU time for more than 4 seconds, its priority is raised to 15 and its time slice is doubled
  – When the time slice expires, thread’s priority and time slice are reverted back to their old values
Performance Analysis of Scheduling Policies

• The set of requests directed at a scheduling policy is called its *workload*
  – First step in performance analysis of a policy is to accurately characterize its typical workload
• Three approaches could be used for performance analysis of scheduling policies:
  – Implementation of a scheduling policy in an OS
  – Simulation
  – Mathematical modeling

Performance Analysis through Implementation

• The scheduling policy to be evaluated is implemented in a real OS that is used in the target operating environment
• The OS receives real user requests; services them using the scheduling policy; and collects data for statistical analysis of the policy’s performance
• Cost is tremendous: coding the algorithm, modifying the kernel data structures, system must be taken down
• This approach is disruptive
  – Disruption can be avoided using virtual machine software
Simulation

- Simulation is achieved by coding scheduling policy and relevant OS functions as a simulator and using a typical workload as its input
  - Workload is a recording of some real workload directed at the OS during a sample period
- Analysis may be repeated with many workloads to eliminate the effect of variations across workloads

![Simulation Diagram]

Figure 7.15 Simulation of a scheduling policy.

Mathematical Modeling

- A mathematical model provides a set of expressions for important performance characteristics such as service times of requests and overhead
- **Queueing analysis**: given statistical distribution of process arrival and statistical distribution of service time, we can compute mean queue length, mean wait time, etc.
  - Assume process arrival follows Poisson distribution, service time follows exponential distribution
    - Probability of an arrival in [0, t] is \( F(t) = 1 - e^{-\alpha t} \), where \( \alpha \) is the mean arrival rate (requests per second)
    - Probability that the service time of a request \( \leq t \) is \( S(t) = 1 - e^{-\omega t} \), where \( \omega \) is the mean execution rate (requests per second)
  - Mean queue length is given by Little's formula:
    - \( L = \alpha \times W \), where \( L \) is the mean queue length and \( W \) is the mean wait time for a request
Mathematical Modeling (continued)

Table 7.7 Summary of Performance Analysis

<table>
<thead>
<tr>
<th>Scheduling policy</th>
<th>Mean wait time for a request with service time = t</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS</td>
<td>$\frac{W_0}{1-\rho}$</td>
</tr>
<tr>
<td>SRN</td>
<td>$\frac{W_0}{1-\rho}$, where $\rho_t = \int_0^t \alpha \cdot y \cdot dS(y)$</td>
</tr>
<tr>
<td>HRN</td>
<td>For small $t$: $W_0 + \frac{\rho^2}{2(1-\rho)} \times \frac{1}{2}$</td>
</tr>
<tr>
<td></td>
<td>For large $t$: $\frac{W_0}{(1-\rho)(1-\rho + \frac{2\rho P_0}{1-\rho})}$</td>
</tr>
<tr>
<td>Round-robin</td>
<td>$\frac{n}{\alpha(1-P_0)} - \frac{1}{\alpha}$, where $P_0 = \frac{1}{\sum_{j=0}^{\infty} \frac{\rho^j \times (\alpha)^j}}$</td>
</tr>
</tbody>
</table>

(P_0 is the probability that no terminal awaits a response)

Note: $W_0 = \int_0^\infty t^2 dF(t)$. For an exponential distribution $F(t) = 1 - e^{-\alpha t}$, it is $\frac{\rho^2}{2\alpha^2}$.

$\rho = \alpha / \omega$

Summary

- Scheduler decides process to service and how long
- Three techniques:
  - Priority-based, reordering of requests, and variation of time slice
- Scheduling can be:
  - Non-preemptive: E.g., FCFS, SRN, HRN
  - Preemptive: E.g., RR, LCN, STG
- OS uses three schedulers: long-term, medium-term, and short-term scheduler
Summary (continued)

• Different scheduling policies
  – Time-sharing:
    • Multilevel adaptive scheduling
    • Fair share scheduling
  – Real-time scheduling
• Performance analysis is used to study and tune performance of scheduling policies