Real-Time Operating Systems Issues

- Example of a real-time capable OS: Solaris.
  S. Khanna, M. Sebree, J. Zolnowsky.
  "Realtime Scheduling in SunOS 5.0". USENIX - Winter ’92.

- Problems with the design of general-purpose real-time capable OS: Solaris
  "SVR4 UNIX Scheduler Unacceptable for Multimedia Applications." NOSSDAV ’93.
  URL: http://www.cs.columbia.edu/~nieh/#publications

Realtime Scheduling in SunOS 5.0

- Requirements for Solaris as a real-time OS:
  - Scheduling of tasks in kernel should be deterministic. Kernel should be free from unbounded priority inversion.
  - Allow for mixed-mode applications: real-time and non-real-time components.
  - Appropriate for multiprocessor machines.
  - Provide standard interface to user, such as System V.

- Historically: unbounded dispatch latency caused by non-preemptible kernel.
  - Solution 1: Well-defined preemption points. (?)
  - Solution 2: Fully synchronize access by kernel code to kernel data structures.
    - Reduces set of non-preemptible portions in kernel.
    - Kernel is multithreaded.
Scheduling Classes

- **Time-Sharing class:**
  - round robin scheduling.
- **Sys class:**
  - fixed priority scheduling,
  - not accessible by the user.
- **Real-Time class:**
  - fixed priority scheduling.
- **priocntl(2)**
  - Change scheduling class or other scheduling parameters.

Scheduling

- State of thread: blocked, runnable, executing
- Scheduling operations (operations on dispatch queue) are protected by single spin lock schedlock.
- Variables per processor:
  - cpu_thread: thread curr. executing
  - cpu_dispthread: thread last sched. for disp
  - cpuidle: special idle thread
  - cpu_runrun: user-level preemption
  - cpu_kprunrun: kernel-level preemption
  - cpu_chosenlevel: next thread to preempt

[Diagram of scheduling classes and states]
Operations on Dispatch Queues

- `setfrontdq()` — put thread in dispatch queue
- `setbackdq()` — (when thread is preempted)
- `cpu_choose()` — find CPU on which runnable thread might be dispatched
- `cpu_surrender()` — give up CPU when priority is lowered
- `disp()` — select a thread for execution from the dispatch queue (used by `swtch`)
- `swtch()` — select highest-priority thread for execution if none is found, returns idle thread modifies many per-processor variables
- `kpreempt()` — attempt to preempt kernel
- `kpreempt_disable()` — disable preemption for critical interval
- `kpreempt_enable()` — reenable preemption

Priority Inversion

- Priority inversion happens due to
  - non-preemptable portions
  - access to synchronization objects
  - “hidden scheduling”
- Synchronization Objects (mutex, r/w locks)
  - Solution: basic priority-inheritance protocol
- Hidden Scheduling
  - Work done asynchronously in kernel on behalf of threads without regard to their priority.
  - Example: streams processing
  - Example: timeouts done at lowest interrupt level
  - Solution: Move this code into kernel threads running at sys priority level.
Priority Inheritance

- Primitives:
  - `pi_willto(thread)` impose priority of argument thread onto all threads that block it, directly or indirectly
  - `pi_waive()` release priority inheritance

- The function `pi_willto()` is called after the thread has been put to sleep in the queue associated with the synchronization object. The information about the synchronization object can therefore be recovered.

- Priority inheritance for readers/writers locks:
  - when writer owns the lock: no problem
  - when readers own the lock:
    - potentially many "owners"; not practical to keep pointer from resource to every thread that owns it
    - Solution: define a single "owner-of-record", which is only thread that inherits priority.

Applicability of SunOS 5.0 for Multimedia Applications

"SVR4 UNIX Scheduler Unacceptable for Multimedia Applications." NOSSDAV '93.
URL: http://www.cs.columbia.edu/~nieh/~publications

- Objectives of real-time OS for general-purpose workstations

  - Provide real-time guarantees without reducing general capabilities of workstations

  - Manage resources so that other applications can operate correctly.

  - SunOS 5.0 (SVR4) provides real-time static-priority scheduler.

- Question: How well are resources managed?
Experimental Evaluation: Overview

- Platform
  - Sun Sparc10
  - Solaris 2.2
  - Scheduling classes (RT class, TS class, SYS class)

- Experiment (measurement) criteria:
  - Interactive:
    - minize average and variance between user input and response
    - Typing, cursor motion, mouse selection <= 50 - 150 ms.
  - Continuous media:
    - Minimize difference between average display rate and desired display rate.
    - Minimize variance of display rate.
  - Batch:
    - "Minimize difference between actual time of completion and minimum time of completion when whole machine is dedicated."

Experiment: Workload

- 3 classes of workload

- Interactive: (editors, GUIs)
  - **Typing**: Emulate a user typing, and display characters on the screen.

- Continuous media: (television, teleconference)
  - **Video**: Capture data from digitizer board and display through x-windows server.

- Batch: (compilations, scientific computation)
  - **Make**: Repeatedly fork and wait for small processes to complete.

- Instrumentation of application and system software components does not measurably change the performance.
Experiment: The Baseline

<table>
<thead>
<tr>
<th>Application</th>
<th>Measurement</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typing</td>
<td>Latency between character arrival and rendering to frame buffer</td>
<td>38.5 ms</td>
<td>15.7 ms</td>
</tr>
<tr>
<td>Video</td>
<td>Time between display of successive frames</td>
<td>112 ms</td>
<td>9.75 ms</td>
</tr>
<tr>
<td>Compute</td>
<td>Time to execute one loop iteration</td>
<td>149 ms</td>
<td>6.79 ms</td>
</tr>
</tbody>
</table>

Table: Application Baseline Values

- What is a well-behaved system?
  - Concurrent applications should make some progress
  - No case where system fails to respond to operator input
  - User should exercise wide range of influence over system behavior.

Experiment 1: Run all tasks in RT class

- Window system is no longer accepting input events from mouse or keyboard.
- Command interpreter not permitted to run.
- System blocked by batch-job
  - Identified as I/O intensive interactive job.
  - Gets priority boosts for sleeping.
- Window server develops backlog of service requests. As it works down its queue, it gets identified as compute bound.

- Table entries are relative to baseline (tall is better)
- T: TYPING character latency
- V: time between display of successive frames for VIDEO.
- C: time for one iteration in COMPUTE.
What can the System Administrator do?

- Increase priority of X-Server, decrease priority of batch task
- In addition, decrease priority of VIDEO a bit
- Decrease priority of VIDEO a little bit more.

Play with RT Class

- e) Video in RT
- f) X-server in RT
- g) Video and X-server in RT, P(X)-P(X)
- h) Video and X-server in RT, P(X)-P(Y)
Result: New TS Class

- Removes anomalies of identifying batch jobs as interactive and vice versa.
- Ensures that each process makes steady progress.
- Reduces feedback interval.
- Included in Solaris 2.3.