Advanced Topics in Computer Systems, CS262b

Brewer/Hellerstein

Nemesis OS Architecture

January 25, 2001

Focus of this paper: OS structure for multimedia applications (in addition to traditional apps)

MM apps:

- temporal property: time-critical demands
- informational property: can often tolerate some data loss

Key observation: QoS crosstalk

- Big OS servers/services don't obey temporal properties... They mix resources among many threads/processes and have high shared overhead that affects everyone.
- Interrupt-driven scheduling subverts the "official" scheduling policy; we can't allow interupts to affect the overall scheduling of resources
- All resources matter, not just CPU

Key ideas:

- provide for dynamic alloc of resources to applications (not OS services)
- must get the accounting of resource usage correct (eg. CPU cycles spent on paging)
- feedback to application to adjust its resource usage (consistent vs. predictable)

CPU use managed by scheduler activations

- threads can be resumed or can have an upcall (to the activation)
- thread contexts managed at user level, kernel only provides atomic context switch and uses the user-level data structures
- no kernel threads: only user-level threads with event dispatch (compare with wakeup predicates in exokernel)
- kernel is completely non-blocking and asynchronous (if you don't have threads you can't block). This ensures that scheduling is only done at user-level (within an sdom).
- A form of two-level scheduling

Interrupts are essentially queued as events:

- After arrival, interrupts are masked, which limits the impact interrupts can have on system performance
- In practice, I/O devices are polled. An (unmasked) interrupt just ensures that the driver runs to do the polling...

- Demultiplexing is the most complicated operation a interrupt handler should do (i.e. which event queue?)

Separate control from data flow (path-based)