CPU-Inheritance Scheduling

Bryan Ford & Sai Susarla (*presented by Stefan Kalkowski*)

January 24, 2007

Focus

- multiple scheduling policies
- count CPU usage
- avoid priority inversion
- generalized notion of priority inheritance

- variety of concurrent uses
- control of resource usage
- no policy in kernel
- arbitrary threads act as schedulers

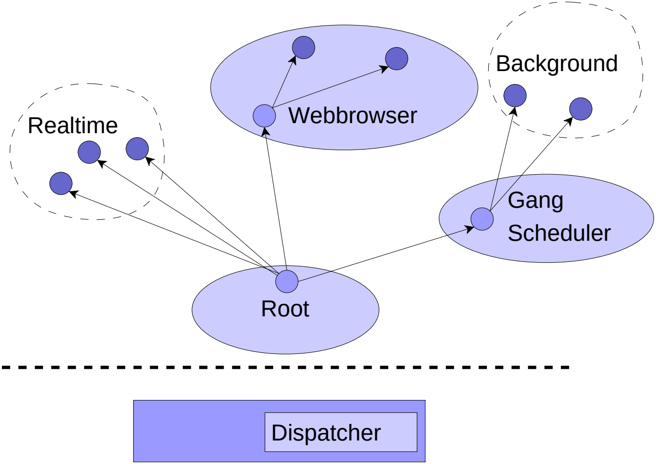
- threads schedule other threads
- every thread might donate its CPU to someone else
- one root scheduler thread foreach CPU
- the dispatcher fields and delivers events
- important events: blocking, ready, out of CPU time

Basic design

- threads schedule other threads
- every thread might donate its CPU to someone else
- one root scheduler thread foreach CPU
- the dispatcher fields and delivers events
- important events: blocking, ready, out of CPU time

- threads schedule other threads
- every thread might donate its CPU to someone else
- one root scheduler thread foreach CPU
- the dispatcher fields and delivers events
- important events: blocking, ready, out of CPU time

- threads schedule other threads
- every thread might donate its CPU to someone else
- one root scheduler thread foreach CPU
- the dispatcher fields and delivers events
- important events: blocking, ready, out of CPU time


Background
Webbrowser
Realtime
Gang
Scheduler
Root
Dispatcher

thread waits for an event:

- obtaining a lock
- a server response
- $\rightarrow$ explicitly or implicitly donate CPU

- `schedule` syscall: donates and waits for event
- 'WAKEUP_ON_BLOCK': normal behaviour
- 'WAKEUP_ON_SWITCH': if target blocks return to grandparent
- 'WAKEUP_ON_CONFLICT': scheduler gets informed only when more then one thread are ready

- `schedule` syscall: donates and waits for event
- 'WAKEUP_ON_BLOCK': normal behaviour
- 'WAKEUP_ON_SWITCH': if target blocks return to grandparent
- 'WAKEUP_ON_CONFLICT': scheduler gets informed only when more then one thread are ready

- `schedule` syscall: donates and waits for event
- 'WAKEUP_ON_BLOCK': normal behaviour
- 'WAKEUP_ON_SWITCH': if target blocks return to grandparent
- 'WAKEUP_ON_CONFLICT': scheduler gets informed only when more then one thread are ready

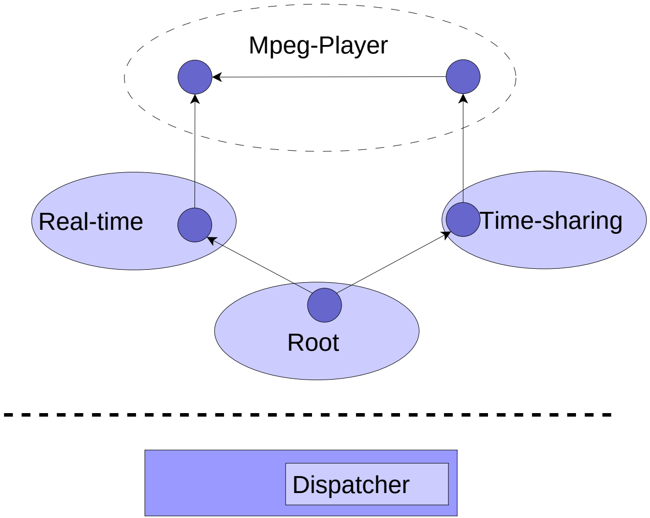
- `schedule` syscall: donates and waits for event
- 'WAKEUP_ON_BLOCK': normal behaviour
- 'WAKEUP_ON_SWITCH': if target blocks return to grandparent
- 'WAKEUP_ON_CONFLICT': scheduler gets informed only when more then one thread are ready

- multithreaded scheduler with shared variables
- scheduler threads listen on one port
- scheduler activations

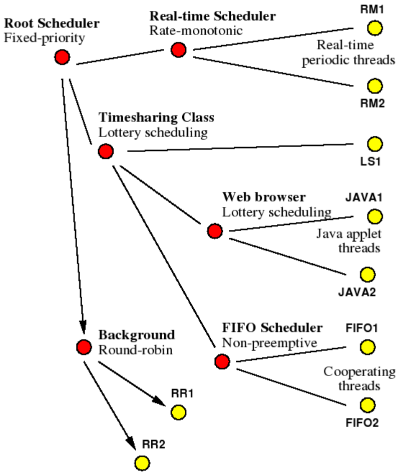
Timing

- schedulers register timeouts
- CPU usage accounting straight forward for root scheduler
- for accurate CPU accounting within the tree, schedulers have to communicate with each other

Multimedia scenario



Test scenario



- load insulation as expected
- avoids priority inversion
- 100% increase of context switches