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

Flexible scheduling
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
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
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
Scheduling hierarchy

Root

Webbrowser

Realtime

Dispatcher

Gang Scheduler

Background

CPU-Inheritance Scheduling
thread waits for an event:

- obtaining a lock
- a server response
- → 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 parent
  • 'WAKEUP_ON_CONFLICT': scheduler gets informed only when more than 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 than 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
Multiprocessors

- multithreaded scheduler with shared variables
- scheduler threads listen on one port
- scheduler activations
• schedulers register timeouts
• CPU usage accounting straightforward for root scheduler
• for accurate CPU accounting within the tree, schedulers have to communicate with each other
Multimedia scenario

Motivation

The Design

Results

Threads with multiple scheduling policies

Dispatcher

Root

Time-sharing

Real-time

Mpeg-Player

Dispatcher (presented by Stefan Kalkowski)
Test scenario

CPU-Inheritance Scheduling

Root Scheduler
Fixed-priority

Real-time Scheduler
Rate-monotonic

RM1
Real-time periodic threads

RM2

Timesharing Class
Lottery scheduling

LS1

Web browser
Lottery scheduling

JAVA1
Java applet threads

JAVA2

Background
Round-robin

RR1

RR2

FIFO Scheduler
Non-preemptive

FIFO1
Cooperating threads

FIFO2

Bryan Ford & Sai Susarla (presented by Stefan Kalkowski)
- load insulation as expected
- avoids priority inversion
- 100% increase of context switches