HLS: A Framework for Composing Soft Real-Time Schedulers

John Regehr – University of Utah
John A. Stankovic – University of Virginia
Dec. 4, 2001

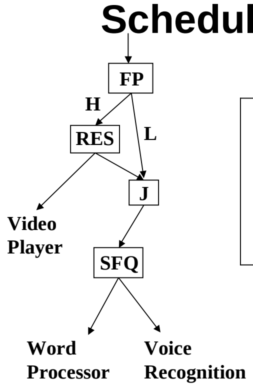
1

Motivation

- People use general-purpose OSs (GPOSs) for many kinds of tasks
  - e.g. Unix, Windows, MacOS variants
  - Compatibility, commodity, convenience
- Applications have diverse scheduling requirements
  - Time-sharing, soft RT, hierarchical isolation, gang scheduling, …
- Schedulers are inflexible
  - Hierarchical scheduling is a solution

2

HLS: Hierarchical Loadable Schedulers



FP:	Fixed Priority
RES:	CPU Reservation
J:	Join
SFQ:	Start-Time Fair Queuing

3

Research Questions

- How to reason about a hierarchical composition of schedulers?
- What novel uses are there?
- Can efficient run-time support for HLS be developed?

4

Contributions

- System of guarantees that permits reasoning about hierarchies
- Build complex behaviors using simple schedulers as components
- Novel implementation using generalization of scheduler activations
  - Runs in Windows 2000 kernel

5

Outline

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion

6

Guarantee

- **Definition:**
  - Ongoing lower (and possibly upper) bound on CPU allocation over time
- **Goals:**
  - Formally describe useful classes of schedules
  - Permit schedules to be reasoned about
- **Syntax:**
  - TYPE *p1 p2 …*

Using Guarantees

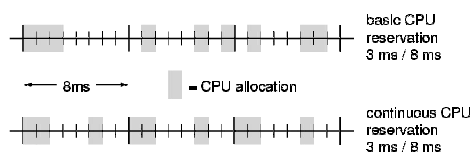
- **Approach: label hierarchy edges with guarantees**
- **Basis step: known label for edge leading to root of hierarchy**
- **Induction step:**
  - Each scheduler requires and provides guarantees
  - Guarantees can be rewritten

Example Guarantees

- **100% of a CPU: ALL**
- **Strictly best-effort scheduling: NULL**
- **Proportional share:**
  - PS *s*, PSBE *s* δ
- **CPU Reservations:**
  - RESBS *x y*, RESBH *x y*
  - RESCS *x y*, RESCH *x y*

CPU Reservation Guarantees

- **Hard / Soft:**
  - "Hard CPU reservation" ≠ hard real-time
  - Soft reservations guarantee a lower bound
  - Hard reservations also guarantee an upper bound
- **Basic / Continuous:**



Guarantee Conversion by Schedulers

- **Schedulers require and provide guarantees**
  - SFQ: PSBE → PSBE⁺
  - Rez: ALL → RESBH⁺
- **Schedulers determine if specific guarantees can be provided**
  - ALL → RESBH 5 10, RESBH 25 100
    - ▯ EDF-based reservation scheduler
    - x Naïve rate monotonic reservation scheduler

Selected Conversions by Schedulers

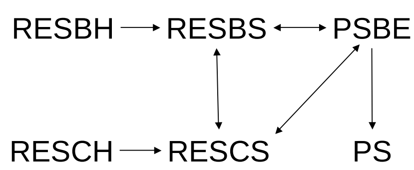
Scheduler	Conversions
Fixed Priority	any → any, NULL⁺
SFQ	PSBE → PSBE⁺, PS → PS⁺
EEVDF	ALL → PSBE⁺
Lottery, Stride	PS → PS⁺
Rialto, Rialto/NT	ALL → RESCS⁺
Rez, CBS	ALL → RESBH⁺
Linux/RT	ALL → RESBS⁺, RESBH⁺
Time Sharing	NULL → NULL⁺

- **Full table contains 23 schedulers**

Guarantee Conversion by Rewrite Rules

- A → B means:
  - Schedule satisfying definition of A also satisfies definition of B
- Trivial examples:
  - PSBE *s* δ → PS *s*
  - RESBH *x y* → RESBS *x y*
- Non-trivial examples:
  - RESBS *x y* → RESCS *x (2y-x+c)* for any $c \geq 0$
  - RESCS *x y* → PSBE $(^x/_y)$ $^x/_y$ $(y\text{-}x)$

Rewrite Rule Overview

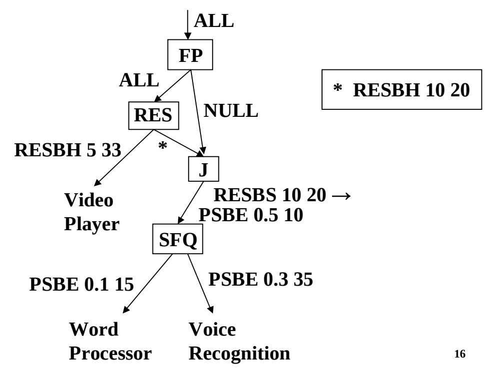


More Rewrite Rules

ALL	T	F	T	F	T	T	T	T
RESBH	F	T	T	F	T	T	T	T
RESBS	F	T	T	T	T	T	T	T
RESCH	F	T	T	T	T	T	T	T
RESCS	F	F	T	F	T	T	T	T
PSBE	F	F	T	F	T	T	T	T
PS	F	F	F	F	F	F	T	T
NULL	F	F	F	F	F	F	F	T
→	ALL	RESBH	RESBS	RESCH	RESCS	PSBE	PS	NULL

T = rewrite rule exists
F = rewrite rule does not exist

Guarantees in Action



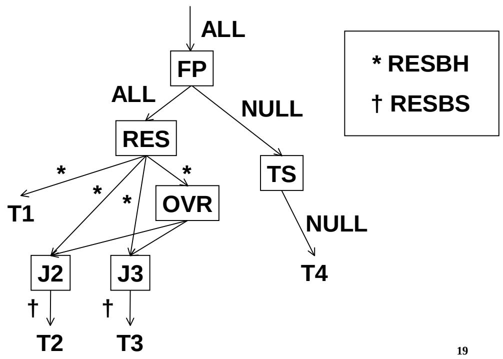
Outline

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion

Example: CPU Service Classes

- Support tasks whose WCET >> average case execution time
- Each task has a CPU reservation
- In addition, tasks share an overrun partition
- Can implement monolithically, or…

CPU Service Classes in HLS



19

Other Complex Behaviors

- Rialto:
  - CPU reservations for groups of threads, RR for indiv. threads
- Portable Resource Kernel:
  - Hard and soft CPU reservations
- Benefits:
  - Little or no coding required
  - Component-based schedulers easy to understand
  - Behaviors are not hardwired

20

Outline

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion

21

Runtime Overview

- Key difference between hierarchical and non-hierarchical schedulers: Revocation
- Explicit notifications
  - Request, release
  - Grant, revoke
- Runtime invariant: schedulers always know number of physical processors they control
  - Permits informed decisions

22

HLS and Scheduler Implementation

- HLS runs in Windows 2000 kernel
  - Added ~3100 lines of code
- Loadable schedulers:
  - CPU reservation, proportional share, join, time sharing / fixed priority
  - A representative set of schedulers, but not a complete one
- Implemented CPU reservations in about two days, PS scheduler in a few hours

23

Performance

- Test machine is a 500MHz Pentium III
- Most mode change operations run in less than 40µs
  - Create / destroy scheduler instance, begin / end CPU reservation, etc.
- Median context switch time
  - Unmodified Windows 2000: 7.1µs
  - HLS time-sharing scheduler: 11.7µs
- Many opportunities for optimization

24

Outline

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion

25

How to Deploy HLS

- Put HLS into a multimedia OS – Windows XP or Linux
- By default:
  - Support interactive, batch, and multimedia applications for a single user
- However, also include
  - Library of useful schedulers and API for composing them
  - API for implementing new schedulers

26

Related Work

- CPU inheritance scheduling [Ford and Susarla 96]
- Hierarchical start-time fair queuing [Goyal et al. 96]
- EDF-based scheduler composition
  - Open environment for real-time applications [Deng et al. 99]
  - BSS-I and PShED [Lipari et al. 00]
- Static and bounded-delay partition models [Mok et al. 00]

27

Conclusion

- Possible to reason about hierarchical composition of soft real-time schedulers
- HLS enables:
  - Complex schedulers to be composed from simple components
  - New schedulers to be developed more easily
- HLS is implemented and performs well

28

The End

- More info and papers: http://www.cs.utah.edu/~regehr/
- Let's talk…

29