**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

---

**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?

---

**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

---

**Outline**

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion
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 $s\delta$
- 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:
  - Hard CPU reservation: PPRE
  - CPU allocation over time
  - CPU reservation: PPRE

Guarantee Conversion by Schedulers

- Schedulers require and provide guarantees
  - SFQ: PSBE $\rightarrow$ PSBE
  - Rez: ALL $\rightarrow$ RESBH
- Schedulers determine if specific guarantees can be provided
  - ALL $\rightarrow$ RESBH 5 10, RESBH 25 100
    - EDF-based reservation scheduler
  - Naive rate monotonic reservation scheduler

Selected Conversions by Schedulers

<table>
<thead>
<tr>
<th>Scheduler</th>
<th>Conversions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Priority</td>
<td>any $\rightarrow$ any, NULL*</td>
</tr>
<tr>
<td>SFQ</td>
<td>PSBE $\rightarrow$ PSBE*, PS $\rightarrow$ PS*</td>
</tr>
<tr>
<td>EEVDF</td>
<td>ALL $\rightarrow$ PSBE*</td>
</tr>
<tr>
<td>Lottery, Stride</td>
<td>PS $\rightarrow$ PS*</td>
</tr>
<tr>
<td>Rialto, Rialto/NT</td>
<td>ALL $\rightarrow$ RESCS*</td>
</tr>
<tr>
<td>Rez, CBS</td>
<td>ALL $\rightarrow$ RESBH+</td>
</tr>
<tr>
<td>Linux/RT</td>
<td>ALL $\rightarrow$ RESBS*, RESBH*</td>
</tr>
<tr>
<td>Time Sharing</td>
<td>NULL $\rightarrow$ NULL*</td>
</tr>
</tbody>
</table>

- 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 \( s \Delta \rightarrow PS s \)
  - RESBH \( x \rightarrow \) RESBS \( x \rightarrow y \)
- **Non-trivial examples:**
  - RESBS \( x \rightarrow \) RESCS \( x \rightarrow (2y-x+c) \) for any \( c \geq 0 \)
  - RESCS \( x \rightarrow \) PSBE \( \left(\frac{x}{y}\right) \rightarrow \left(\frac{x}{y}\right) \) \( (y-x) \)

More Rewrite Rules

<table>
<thead>
<tr>
<th>ALL</th>
<th>T</th>
<th>F</th>
<th>T</th>
<th>F</th>
<th>T</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESBH</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>RESBS</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>RESCH</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>RESCS</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>PSBE</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>PS</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

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

Rewrite Rule Overview

- RESBH → RESBS → PSBE
- RESCH → RESCS → PS

Guarantees in Action

- 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…

Outline

- Motivation and Approach
- Guarantees
- Building Complex Behaviors
- Runtime Support
- Conclusion
**CPU Service Classes in HLS**

- **ALL**
- **FP**
- **NULL**
- **RES**
- **TS**
- **OVR**
- **T1**
- **T2**
- **T3**
- **T4**

**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

**Outline**

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

**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

**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

**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
Outline

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

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

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]

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

The End

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