Dynamic Resource Allocation for Priority Processing
– Master Project –

Martijn van den Heuvel
m.m.h.p.v.d.heuvel@student.tue.nl

Systems Architecture and Networking (SAN)
Department of Mathematics and Computer Science
Eindhoven University of Technology
the Netherlands

23 July 2009
1 Introduction
   - Scalable Video Algorithms
   - Platform
   - Real-time Systems
   - Priority Processing

2 Resource Management Mechanisms
   - Preliminary Termination
   - Processor Allocation

3 Simulation Environment

4 Simulation Results
   - Preliminary Termination
   - Processor Allocation

5 Conclusions
Trade-off: Quality versus Resources

- Reuse software modules
- Cost-effective
- Time-to-market (Porting to new platforms)

By courtesy of Hentschel et. al.
Scalable Video Algorithms Motivation

Trade-off: Quality versus Resources

- Reuse software modules
- Cost-effective
- Time-to-market (Porting to new platforms)

By courtesy of Hentschel et. al.

Demo …
Platform Definition

Definition

A *platform*, which is capable to run software components, is defined by:

1. hardware (resources)
2. programming language
3. operating system (OS)
4. runtime-libraries (provided by - or built on top of - OS)

Note: A resource can also be a software entity!

(We especially consider the processor)
A *platform*, which is capable to run software components, is defined by:

1. hardware (resources)
2. programming language
3. operating system (OS)
4. runtime-libraries (provided by - or built on top of - OS)

Note: A *resource* can also be a software entity! (We especially consider the processor)
Hardware Architectures Overview

Current trend

Desktop PC / Laptop
General Purpose

Signal Processing Acceleration
DSP

Telephone chips
ASP

Flexibility
efficiency

By example of: H. Corporaal, B. Mesman
1. Abstraction
   - provide generic concepts;
   - handle complexity.

2. Virtualization
   - same abstraction for multiple underlying systems;
   - each user is provided a dedicated platform (sharing of platform).

3. Resource management
   - sharing, protection of resources;
   - optimize performance of resource usage.
### Operating System Support

1. **Abstraction**
   - provide generic concepts;
   - handle complexity.

2. **Virtualization**
   - same abstraction for multiple underlying systems;
   - each user is provided a dedicated platform (sharing of platform).

3. **Resource management**
   - sharing, protection of resources;
   - optimize performance of resource usage.

---

**Research subject:** Resource Management Mechanisms
Real-time Systems

Definition

A *real-time system* is a system which has to fulfill:

1. Functional correctness
2. **Timeliness correctness**

![Diagram](chart.png)

courtesy by Dr. Isovic
A *real-time system* is a system which has to fulfill:

1. Functional correctness
2. **Timeliness correctness**

Requires Analysis!
**Definition**

A *task* is a sequence of actions that must be performed in reaction to an event.

**Definition**

A *job* is an instantiation of a task.
Real-time Task Model

Definition
A task is a sequence of actions that must be performed in reaction to an event.

Definition
A job is an instantiation of a task.

Properties of a real-time task:
1. Timing parameters known upfront, (i.e.: computation time, deadline, period);
2. Assume worst-case execution time (WCET) near to average-case execution time;
3. priority assignment.
Policy versus Mechanism

Definition

A *policy* defines how a system should behave (abstract).

Definition

A *mechanism* provides the instrument to implement the policy.
Policy versus Mechanism

**Definition**

A *policy* defines how a system should behave (abstract).

**Definition**

A *mechanism* provides the instrument to implement the policy.

**Example:** processor sharing (scheduling)

- **Policy:**

  **Round Robin**
  
  Assign time-slots, $\Delta t_s$, alternating over tasks
  
  \[ \cdots \quad | \quad \Delta t_s \quad | \quad \cdots \]

  
  \[
  \begin{array}{ccc}
  \text{task 0} & \text{task 1} & \text{task 2}
  \end{array}
  \]

- **Mechanism:** priority based scheduling and pre-emption
Definition

*Fixed Priority Pre-emptive Scheduling* means:

1. highest priority ready task will always execute
2. during runtime the scheduler can *pre-empt* a lower priority task in favor of a higher priority task
Video Processing Characteristics

- Periodic behavior with corresponding deadlines
- Data-dependent, highly Fluctuating Load

WCET **NOT** near to average case

Classical real-time approach leads to under-utilization of (processor) resources.
1. **Basic**: simple and fast output at low quality;

2. **Analysis**: Sort video content in order of importance;

3. **Enhance**: Process video content according to sorted order;

Termination is allowed after a basic output is available.
**Definition**

*Overload: A task’s computation time exceeds the deadline*

\begin{align*}
D &= T \\
\text{Period } (T)
\end{align*}
**Definition**

**Overload:** A task’s computation time exceeds the deadline

\[ D = T \]

**Required:** Mechanism for preliminary termination
Definition

Gain-time: Assigned, but unused processor resources becoming available for other tasks

\[ D = T \]
Priority Processing - Gain-time

Definition

Gain-time: Assigned, but unused processor resources becoming available for other tasks

\[
\begin{align*}
& t_a \\
& \text{Gain-time} \\
& t_d \\
& \text{basic} \quad \text{scalable} \\
& \text{Period (}T\text{)} \\
& D = T
\end{align*}
\]

Desired: Mechanism allowing gain-time consumption
Multiple independent algorithms share a single processor.

- Divide period in fixed-size quanta, e.g. time-slots $\Delta t_s$.
- Decision Scheduler divides time-slots:

$$\text{Algorithm 1} \quad \text{Algorithm 2} \quad \Delta t_s \quad \text{period}(T)$$

Hence: Dynamic Allocation.
Multiple independent algorithms share a single processor.

Divide period in fixed-size quanta, e.g. time-slots $\Delta t_s$.

Decision Scheduler divides time-slots:

Hence: Dynamic Allocation
Multiple independent algorithms share a single processor
Divide period in fixed-size quanta, e.g. time-slots $\Delta t_s$
Decision Scheduler divides time-slots:

Required: Monitoring of time and progress on time-slot scale
Outline

1. Introduction
   - Scalable Video Algorithms
   - Platform
   - Real-time Systems
   - Priority Processing

2. Resource Management Mechanisms
   - Preliminary Termination
   - Processor Allocation

3. Simulation Environment

4. Simulation Results
   - Preliminary Termination
   - Processor Allocation

5. Conclusions
• Decision scheduler aims at maximizing total progress of algorithms
• Decision Scheduler defines a scheduling policy, e.g. based on Reinforcement Learning or Round Robin
• Focus is on mechanisms to support (optimize) the decision scheduler
Priority Processing Architecture (2/2)

Distribution of Responsibilities:

<table>
<thead>
<tr>
<th>Application Specific (Policy)</th>
<th>System Support (Mechanism)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll-forward when deadline reached</td>
<td>Preliminary termination</td>
</tr>
<tr>
<td>Resource distribution (Reinforcement Learning policy)</td>
<td>Allocation of processor (Scheduling)</td>
</tr>
<tr>
<td>Monitor Progress values</td>
<td>Account consumed time (time-slots)</td>
</tr>
</tbody>
</table>
Goal:

Share processor resources among competing priority processing algorithms as efficient as possible.

Compare performance of different implementations for the mechanisms:

1. Preliminary Termination
2. Processor Allocation
Resource Management Mechanisms

Goal:

Share processor resources among competing priority processing algorithms as efficient as possible.

Compare performance of different implementations for the mechanisms:

1. Preliminary Termination
2. Processor Allocation

Note: Monitoring is missing and left for discussion.
Preliminary Termination

Terminate all jobs when deadline is reached:

1. **Cooperative termination (polling):**
   - Decision Scheduler sets a flag when the deadline expires;
   - Algorithms check on regular intervals for deadline expiration.

   ```
   while (!endOfFrame and !deadlineExpired)
   processNextBlock();
   ```
Preliminary Termination

Terminate all jobs when deadline is reached:

1. **Cooperative termination (polling):**
   - Decision Scheduler sets a flag when the deadline expires;
   - Algorithms check on regular intervals for deadline expiration.

   ```
   while (!endOfFrame and !deadlineExpired)
   processNextBlock();
   ```

2. **A-synchronous signalling:**
   - Decision Scheduler sends a signal causing all algorithms to preliminary terminate jobs

   ![Diagram of signal handling](image-url)
Dynamically Allocate Processor Resources:

1. **Suspend-resume tasks:**
   Suspend only in favor of others!
Dynamically Allocate Processor Resources:

1. **Suspend-resume tasks:**
   Suspend only in favor of others!

2. **Manipulate priorities:**
   Substitute:
   - suspend $\rightarrow$ assign low priority
   - resume $\rightarrow$ assign high priority

Low priority tasks can use gain-time.
Priority Processing Applications are prototyped using:

- Matlab-Simulink (version 2007b)
- Microsoft Windows XP:
  - use Fixed Priority Scheduler (FPS)
  - administrator privileges required
Priority Processing Applications are prototyped using:

- Matlab-Simulink (version 2007b)
- Microsoft Windows XP:
  - use Fixed Priority Scheduler (FPS)
  - administrator privileges required
- General Purpose Multi-core machine:
  - Scalable Video Algorithms
  - Decision Scheduler
  - Matlab environment
    (extract results from simulation)
Priority Processing Applications are prototyped using:

- Matlab-Simulink (version 2007b)
- Microsoft Windows XP:
  - use Fixed Priority Scheduler (FPS)
  - administrator privileges required
- General Purpose Multi-core machine:
  - Scalable Video Algorithms
  - Decision Scheduler
  - Matlab environment
    (extract results from simulation)
- Video sequences from VQEG
Outline

1. Introduction
   - Scalable Video Algorithms
   - Platform
   - Real-time Systems
   - Priority Processing

2. Resource Management Mechanisms
   - Preliminary Termination
   - Processor Allocation

3. Simulation Environment

4. Simulation Results
   - Preliminary Termination
   - Processor Allocation

5. Conclusions
Computational Overhead of polling-based mechanisms:

Polling overhead for scalable deinterlacer (VQEG source 6)

Relative Computation Time (% add. time)
Number of processed frames

Conclusion:
Polling causes small overhead on general purpose machines.
Computational Overhead of polling-based mechanisms:

![Graph showing polling overhead for scalable deinterlacer (VQEG source 6)]

**Conclusion:** Polling causes small overhead on general purpose machines.
Polling based termination latencies:

Termination latency for deinterlacer (VQEG src6)

Latency (µs) vs Number of processed frames

Block-based polling

Pixel-based polling

Conclusion:
Lower bound on latency due to OS overhead.

Martijn van den Heuvel (TU/e, SAN)
Dynamic Resource Allocation
23 July 2009 28 / 34
Polling based termination latencies:

**Conclusion:** Lowerbound on latency due to OS overhead.
Signalling based termination latencies:

- Implemented in Linux (not in Windows)
- Unpredictable latencies

**Conclusion:** Polling is the most general solution.
Processor Allocation - Gain-time

How suitable is Priority Processing for gain-time consumption?

1. Use Round Robin policy within Decision Scheduler
2. Non-optimal scheduling (leaving gain-time)
Processor Allocation - Gain-time

How suitable is Priority Processing for gain-time consumption?

1. Use Round Robin policy within Decision Scheduler
2. Non-optimal scheduling (leaving gain-time)

![Graph showing Priority Manipulation vs. Suspend/Resume using RR]

### Conclusion:
Priority manipulation allows gain-time consumption

Martijn van den Heuvel (TU/e, SAN)
Processor Allocation - Gain-time

How suitable is Priority Processing for gain-time consumption?

1. Use Round Robin policy within Decision Scheduler
2. Non-optimal scheduling (leaving gain-time)

Conclusion: Priority manipulation allows gain-time consumption
Optimized Control

1. Use block-based polling
2. Use Windows FPS
3. Reduced context-switching
4. Allow gain-time consumption

**Graph:**
- *Y-axis:* Relative Gain (% processed blocks)
- *X-axis:* Period (ms)
- *Legend:*
  - Enhancement Filter
  - Deinterlacer

Conclusion:
Significant improvement by reducing control overhead
Optimized Control

1. Use block-based polling
2. Use Windows FPS
3. Reduced context-switching
4. Allow gain-time consumption

Conclusion: Significant improvement by reducing control overhead
Outline

1 Introduction
   - Scalable Video Algorithms
   - Platform
   - Real-time Systems
   - Priority Processing

2 Resource Management Mechanisms
   - Preliminary Termination
   - Processor Allocation

3 Simulation Environment

4 Simulation Results
   - Preliminary Termination
   - Processor Allocation

5 Conclusions
Conclusions

- Identification of 3 mechanisms for dynamic resource allocation:
  1. Preliminary termination:
     - Most platforms, including Windows, do not support signalling
     - Polling is preferred (trade-off granularity)
  2. Processor allocation:
     - Reduced context-switching
     - Priority manipulation allows gain-time consumption
  3. Monitoring:
     - On a time-slot scale
     - Relies on availability of high-resolution timers

Open challenges:

1. Optimize processor utilization:
   - Overhead caused by interference of simulation environment
   - Optimize memory management
2. Mapping on embedded platform (real-time environment)
Conclusions

- Identification of 3 mechanisms for dynamic resource allocation:
  1. Preliminary termination:
     - Most platforms, including Windows, do not support signalling
     - Polling is preferred (trade-off granularity)
  2. Processor allocation:
     - Reduced context-switching
     - Priority manipulation allows gain-time consumption

Open challenges:

1. Optimize processor utilization:
   - Overhead caused by interference of simulation environment
   - Optimize memory management
2. Mapping on embedded platform (real-time environment)
Conclusions

- Identification of 3 mechanisms for dynamic resource allocation:
  1. Preliminary termination:
     - Most platforms, including Windows, do not support signalling
     - Polling is preferred (trade-off granularity)
  2. Processor allocation:
     - Reduced context-switching
     - Priority manipulation allows gain-time consumption
  3. Monitoring:
     - On a time-slot scale
     - Relies on availability of high-resolution timers
Conclusions

- Identification of 3 mechanisms for dynamic resource allocation:
  1. Preliminary termination:
     - Most platforms, including Windows, do not support signalling
     - Polling is preferred (trade-off granularity)
  2. Processor allocation:
     - Reduced context-switching
     - Priority manipulation allows gain-time consumption
  3. Monitoring:
     - On a time-slot scale
     - Relies on availability of high-resolution timers

- Open challenges:
  1. Optimize processor utilization:
     - Overhead caused by interference of simulation environment
     - Optimize memory management
  2. Mapping on embedded platform (real-time environment)
References

L. F. Bic and A. C. Shaw.  
*Operating Systems Principles.*  

C. Hentschel and S. Schiemenz.  
Priority-processing for optimized real-time performance with limited processing resources.  

S. Schiemenz.  
Echtzeitsteuerung von skalierbaren Priority-Processing Algorithmen.  