What’s Keeping Hard Real-Time Scheduling from Being a Mainstream Technology in the Embedded Multiprocessing Domain Space

Daniel M. Lorts
University of Texas – Dallas and Mercury Computer Systems, Inc.

HPEC - September 2002

The Ultimate Performance Machine
The choices and assumptions made in the development of real-time systems affect many areas.

In this research, we look at six individual, but closely related components of a system architecture.

To make the scheduling problem simpler, various assumptions, or boundary conditions, in one or more of the models are typically made.

Why?
- No-hard/complete problem
- Single semester projects
- Limited tenureship of research
- Focused interest/purpose
### System model assumptions
- Heterogeneous processing assets
- High-level processing capabilities
- Fictitious architectures and topologies
- Assets fully connected

### Communication model assumptions
- Negligible communication costs
- Uniform communication costs
- Non-contentious communications
- No priority discernment
Multi-Level System Graph

S = (R, L)  

\[ R = (r_1, r_2, \ldots, r_N) \]  

N resources of system  

\[ L = \{l_{<,c>} | \subset R \land c > 0 \land \text{Adj}(\cdot) = 1 \} \]  

- \( l \) represents a link connecting two resources  
- \( \subset \) is a subset of the resources of system \( (R) \)  
- \( c \) represents the number of links between the resources  
- \( \text{Adj}() \) is a binary function testing for presence of links  

- **Advantages of the MGS:**  
  - Multi-path capability between resources  
  - Total # of communication links bounded by I/O ports  
  - Ability to model all multiprocessing topologies  
  - Scalable to account for resources that have multiple functional units  

\[ r_1 = \{c_1, c_2, \ldots, c_M\} \]  

where \( c_j \) is a unique capability of resource \( r \)
An MSG Example

S=(R,L) where
R={R1,R2,R3,R4}
and L={l<à1,c1>, l<à2,c2>, l<à3,c3>, l<à4,c4>}

with à1 = {R1} and c1=1
à2 = {R1,R2} and c2=1
à3 = {R2,R4} and c3=2
à4 = {R2,R3,R4} and c4=1

Alternate Representation

The system model can also be defined mathematically within a table. Each cell represents the paths that exist between each resource: l<id,count>. 

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>l(4,1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>l(1,1)</td>
<td>l(2,1)</td>
<td>l(2,1)</td>
<td>l(3,2)</td>
</tr>
<tr>
<td>R3</td>
<td></td>
<td>l(2,1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>l(2,1)</td>
<td>l(3,2)</td>
<td>l(2,1)</td>
<td></td>
</tr>
</tbody>
</table>
Dynamic scheduling

- Transfer policy
- Selection policy
- Location policy
- Information policy

Heuristics include:

- Heavy Node First (HNF)
- Critical Path Method (CPM)
- Weighted Length (WL)
- Earliest Deadline First (EDF)
- Least Laxity First (LLF)
Dynamic Framework

- Allocation stage: statically assigning processes to resources
- Scheduling stage: dynamically based on allocation and application
- Provides run-time analysis of loading and balance
- Scalable solution for all multiprocessing system applications
- Possible multicomputer processor configurations
  - Round-robin/next available
  - Single parallel cluster
  - Pipeline of parallel clusters
  - Hybrid
Non-realistic program models (DAGs)

Task model limited
- Uniform temporal metrics
- Preemptability
- Entry points into nodes
- Typically unary dependencies
- Limited methods of prioritization
- Acyclic models limited

Task Variables
- Computational times
- Communications times
- Deadlines (laxity)
- Precedence
  - Number of children
  - Number of parents
Other technological issues

- Compiler optimization techniques
- Multifunctional resources
- Size, weight, and power considerations
Fault Modeling

CAUSES
- Specification
- Implementation
- Externals
- Defects

TYPES
- Fault Avoidance
- Fault Masking
- Hardware
- Software

ERRORS
- Fault Tolerance

FAILURES

Fault model
- Too simplistic
- Single fault assumption
- Fault isolated to task or processor
- Limited recovery techniques
- Inconsistent QoS issues

Drivers
- Visibility
- Cost
- Affects
The Scheduling Framework

A Real-Time Hybrid Scheduling Framework for Fault Tolerant Polymorphic Computing

Temporal Characteristics Hard, Soft, Fuzzy, Real-Time
Static Plus Dynamic Adaptive
Structured Development
Plan for Detection, Correction, and Recovery
Application Awareness Reconfigurability
Quality of Service (QoS) Runtime Performance Development Efficiency
Framework Details

- **Structured development**
  - Provides foundation
  - Validated by mapping known architecture

- **Hybrid scheduling**
  - Focal point of research
  - Static and dynamic approaches

- **Real-Time**
  - Formal temporal methods

- **Fault tolerance**
  - Dynamic detection, correction, and recovery

- **Polymorphism**
  - Reactive environments
  - Efficient ‘morphing’

- **Computing**
  - Quality of Service (QoS)
  - Usability and generality