An Architecture to Enable Machine-Learning-Based Task Migration for Multi-Core Real-Time Systems

Octavio Delgadillo
Team

Octavio Ivan Delgado Ruiz  
Fortiss GmbH
ruiz@fortiss.org

Bernhard Blieseniger  
Fortiss GmbH
bliening@fortiss.org

Juri Kuhn  
Fortiss GmbH
kuhn@fortiss.org

Prof. Dr. Uwe Baumgarten  
Technical University of Munich
baumgaru@tum.de
Agenda
1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Motivation

• Most innovations in automotive industry are driven by electronic hardware and software developments

• More than 1/3 of total car costs are in the electronic devices (Hainz & Chauhan, 2016) → Potential for optimization
Problem Statement

• Problem: modern cars integrate dozens of electronic control units (ECUs). Many serve single-purpose (inefficient)

• Possible solution: ECU consolidation
• Challenges: Maximization of efficiency. Safety (redundancy, real-time constraints, criticality of tasks)
Proposed Solution

- Enable task migration between different ECUs at runtime
  - Task to ECU distribution planned by performing schedulability analysis
- Traditional (mathematical) approaches are not always feasible
- How about using an approximative and adaptive approach for performing the schedulability analysis?

→ Introduction of Machine-Learning-Based Algorithms
Machine-Learning-Based Schedulability Analysis

<table>
<thead>
<tr>
<th>Disadvantages</th>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inexact (only gives a prediction on empirically derived data)</td>
<td>Potentially faster predictions</td>
</tr>
<tr>
<td>Behavior not fully deterministic (not certifiable)</td>
<td>Potentially achieving higher utilization</td>
</tr>
<tr>
<td></td>
<td>Adaption to uncertainties and system updates possible by learning</td>
</tr>
</tbody>
</table>

- **Challenges:**
  - Integration of ML predictions with real-time platforms.
  - Fast detection / prediction of failures with task distributions.
- **Here presented is an architecture for:**
  - Dynamic distribution of a set of real-time tasks to a set of ECUs
  - Collection of execution data for training machine learning algorithms
Agenda

1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Task Deployment

- Server tool triggers schedulability analysis
  - Generates possible task distributions
  - Monitors and triggers task execution on ECUs

- Communication exchange:
  - Server $\rightarrow$ ECU: Tasks assigned (ID, WCET, period, number of jobs, start/stop)
  - ECU $\rightarrow$ Server: Task status (Completed jobs, missed deadlines, running/finished/failure)
Task Execution

- Preemptive EDF scheduling $\rightarrow$ Optimal utilization
  - Slight overhead for the dynamic deadline calculation
  - Reduced context switch overhead
- EDF policy is optimal for single core processors, and achieves utilization of 1 (see Buttazzo, 2011)
Task Execution Monitoring

- Task deadline misses and jobs completed are counted
- Tasks can be created or removed at runtime
- Tasks can be started, paused or stopped at any time
- Information sent back to central deployment tool
  - Recognizing task or device failures
  - Online learning
Autonomous Task

- Simulation of a robot navigating a labyrinth (server)
- SLAM algorithm and motion control (device)
- Communication with server (device)
- Period of 1s
- Uncertainty in environment and communication gives as a result great variation in execution time. Avg: 210ms, WCET: 550ms
Dummy Tasks

• Filler for achieving high utilization
• Automated generation of tasks (COBRA)
• Periods ranging from 100ms to 10s
• Execution times based on the repeated execution of simple programs
• Introduce a few challenges that are easier to solve with the ML approach
  • Mutex dependencies
  • Slightly inexact WCET values
Agenda

1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Test Setup

FreeRTOS Device 1
- Pathfinder
- Dummy Tasks

FreeRTOS Device 2
- Dummy Tasks

FreeRTOS Device 3
- Dummy Tasks

Linux Server
- NeuroRobotics Platform Docker Container
  - Robot / Environment Simulation
  - ROS Proxy

Task Deployment Tool / ECU Management

SpiNNaker Connection

SpiNNaker
- Schedulability Analysis with Spiking Neural Networks
Schedulability Analysis (ML)

- In this work, a Spiking Neural Network (SNN) was used
- Adaption of other developed algorithms is being worked upon
- SNN developed is simple but good for testing architecture
  - 1 input population (32 neurons), 1 output population (20 neurons)
  - All-to-all STDP synapses
  - 1 teaching population for training
ECU System

- Ultra96 development board
  - Based on Xilinx UltraScale+ MPSoC
- 2 processors:
  - ARM Cortex-A53 (4 cores)
  - ARM Cortex-R5 (2 cores)
- Only active core:
  - Core 0 of A53
- Operating System:
  - FreeRTOS
- Testbed:
  - 3 ECUs
Agenda
1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Results

• All tasks are able to execute and fulfill their goals, unless no feasible schedule is found.

• ECUs can achieve very high utilization: more than 98% of task sets with utilization <0.95 successful, even achieving > 1 (According to WCET).

• Tasks can be moved around freely at any moment between ECUs.

• System reacts in case an ECU fails or if tasks miss deadlines.
Results (contd.)

- SNN achieved an accuracy of 0.85 with 4500 training samples and 1500 test samples
- SNN is capable of online learning
- Other ML algorithms have shown better results, but are still being adapted to work with the architecture here presented

<table>
<thead>
<tr>
<th>True Label</th>
<th>Prediction</th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Successful</td>
<td>Unsuccessful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successful</td>
<td>0.675</td>
<td>0.105</td>
<td></td>
<td>0.780</td>
</tr>
<tr>
<td>Unsuccessful</td>
<td>0.045</td>
<td>0.175</td>
<td></td>
<td>0.220</td>
</tr>
<tr>
<td>Total</td>
<td>0.720</td>
<td>0.280</td>
<td></td>
<td>1.000</td>
</tr>
</tbody>
</table>
Agenda
1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Discussion

- Improvement upon our previous platform
- System allows for real-time behavior being monitored
- System allows for small execution times and periods running along larger ones
- Developed tasks and architecture allow for creation of a very robust task set for training ML
Shortcomings

• Task migration not fully implemented
  • Task code is in every ECU
  • Task context is local to ECU

• Communication overhead not considered
  • Communication not bounded

• Deadlines local to each ECU
  • Overall system synchronization required
Agenda

1. Introduction
2. Implementation
3. Test-Setup and Platform
4. Results and Analysis
5. Discussion
6. Conclusion and Future Work
Conclusion

- We introduce a development platform to enable task migration, with the aim for ECU consolidation
- This work serves as baseline for our further developments
- This work allows for data collection to perform training of the ML algorithms
- This work sets a milestone on our ongoing research
Future Work

- Approach generalization
- Extending to multi-core architectures
- Deployment of task binaries at runtime
- Adaptation and comparison of ML algorithms
Image Sources
Slides 2 & 5: Carmakers, Tech Giants Join Forces in Historic Partnership Against Hackers – autoevolution
Slide 3: Vehicle electronic control unit (ECU) consolidation targets the self-driving autonomous car | VentureOutsource.com
Slide 18: Ultra96 - 96Boards