Migration of Components and Processes as means for dynamic Reconfiguration in Distributed Embedded Real-Time Operating Systems

Sebastian Eckl, Daniel Krefft, Uwe Baumgarten
Technische Universität München
TUM Department of Informatics
Chair of Operating Systems
OSPERT 2017, Dubrovnik, 27.06.2017
Agenda

• Research Environment
• Challenge
• Proposed Solution: Dynamic Reconfiguration
• Operating System Design
  • Partitioning
  • Migration Support
• Implementation (Current State)
• Evaluation (Test Setup)
• Next Steps
• Conclusion
Research Environment

- Future cyber-physical-systems (CPS) as distributed, embedded, real-time systems in a criticality-aware environment

- Key demands
  - Multi-core hardware support
  - Mixed-criticality support
  - Context-sensitive reaction behavior support

- Examples
  - Automotive Systems (e.g. Autonomous Driving, C-ITS)
  - Industrial Automation (e.g. Industry 4.0)
Challenge

- **Vision**: Offer context-sensitive reaction behavior via *Dynamic Reconfiguration* at run-time
- **Tradeoff**: Flexibility vs. Reliability/Safety
- State-of-Art *safety guarantee*: physical separation of software components (spatial & temporal isolation)
- State-of-Art reconfiguration: *Semi-statically configured systems*
  - **Mode-switch** mechanism (run-time substitution of a set of precalculated, offline schedules)
  - Hardware-supported *redundancy* (device switch)
Proposed solution

• Idea: (Online) Dynamic reconfiguration via migration at runtime
  • Migration of software components and processes
  • Between different devices
  • Integration into operating system (OS) layer
• Background: adoption of run-time adaptation concepts from data centers (virtual machine migration)
  • Efficient resource management
  • Fault tolerance/availability
  • Scalability
General Design Assumptions

- Application of (heterogeneous) **COTS hardware**
- **Homogeneous** software-based run-time environment (RTE)
- Real-Time Operating System (**RTOS**)
- **L4 microkernel** based OS (small & minimalist base)
- Migration: **Extension of OS** via
  - Migration decision making
  - Migration execution
Operating System Design

• Premise: Keeping up a system‘s real-time capability
• Implementation in Userland
• Base: **Multi-core platform** (concurrent execution)
  • Goal: Temporal isolation
  • Limitation: **Shared resources** (e.g. system buses, caches)
• Approach
  • **Partitioning**: Mapping of software components of different criticality level to corresponding cores
  • **Migration**: Stepwise evaluation (from migration of lower-critical components to dealing with harder constraints)
• Implementation: L4 Fiasco.OC & Genode OS Framework
Combined management of best-effort, soft and hard real-time tasks.
Background: Future Trends


Physically separated management of best-effort, soft and hard real-time tasks.
Partitioning – Prototypical Development

- UDOO Dual/Quad (NXP i.MX6 Cortex-A9 + Arduino Due-compatible ATMEL Cortex-M3)

http://www.udoo.org/docs/Hardware_&_Accessories/IMX6_And_Sam3X_Communication.html

Sebastian Eckl, Daniel Krefft, Uwe Baumgarten © 06/2017
Migration Process

• Runtime Monitoring
• Global Migration Decision (Analysis, Planning)
• Migration
  • Checkpointing (save all necessary process data)
  • Serialisation (prepare data for transport)
  • Transfer (send data to target platform)
  • De-Serialisation (unpack data on target platform)
  • Restore (restart process on target platform)
  • OPTIONAL: Reorganisation (restore communication paths)
• Local Admission Control
• Synchronisation
• Real-Time Scheduling
Migration - OS Components

• Observer/Controller (migration planning)
  • Runtime Monitoring
  • Admission Control (local online schedulability analysis)
  • Global Planning (global online schedulability analysis)
• Core OS (migration execution)
  • Snapshot (Checkpoint/Restore, De/Serialization)
  • Storage (In-Memory)
  • Networking (Transfer)
  • Synchronisation (Deployment)
  • Real-Time Scheduling (Enforcement)
Core OS – Checkpoint/Restore

• **Periodic** snapshot creation: context switch (scheduler preempt)
• Target: Reducing **duration of memory checkpointing** by
  • Snapshot granularity (full, partial, **incremental** memory dump)
  • Processing implementation
    • Sequential (Single-Core based Stop/Start)
    • **Parallel** (Multi-Core based **Copy-on-Write**)
  • Type of Memory Access (Separated-, Shared-, Redundant-Memory)
• Dedicated **hardware support** (e.g. Co-Processor/FPGA based)
Background: L4 Fiasco.OC & Genode

- [L4 Fiasco.OC](https://l4re.org/doc/l4re_intro.html)
- [Genode](http://genode.org/documentation/general-overview/index)

Sebastian Eckl, Daniel Krefft, Uwe Baumgarten © 06/2017
Hierarchical component structure

- Application_1
- Application_2
- Application_n
- Analysis/Planning
- Broker/Checkpoint/Restore
- genode-Synchronisation
- genode-Monitoring
- int
- core
- genode-dcm0
- genode-parser
- genode-AdmCtrl
- L4

OS
- base/base-foc (SAL)
- Kernel (HAL)
Current Development – Embedded System

• **Operating System**
  • Runtime Monitoring
  • Checkpoint/Restore (Shared Memory)
    • Granularity (Full, Partial, Incremental)
    • Method (Stop/Start, Copy-on-Write)
    • Multi-Core mapping
  • Partitioning (Affinity)
  • Admission Control (Online schedulability analysis)
  • Synchronisation (in idle mode)
  • Real-Time Scheduling (FPP, EDF)
• **Hardware Support** (Porting to ARM Quad-Core platforms)
Open Source – Argos-Research

For more info see
https://github.com/argos-research
Current Development – Hardware support

**Current support:** ARMv7-A (Cortex-A9, Dual-Core)
- Pandaboard ES (TI OMAP 4)
- Digilent Zybo (Xilinx Zynq-7000)
- QEMU (RealView Platform Baseboard Explore) (PBX-A9)

**Future support:** ARMv7-A (Cortex-A9, Quad-Core)
- ODROID U3 (Samsung Exynos 4412)
- Wandboard (NXP i.MX6)
- UDOO Board (NXP i.MX6)
Test Cases – Automotive/Robotics

- Production line (industrial automation)
  -> energy saving scenario

- Autonomous Parking
  -> fault-tolerance scenario (e.g. fail-safe)

Test Setup – Hybrid Simulator
Next Steps – Operating System

- **Migration Execution**
  - (De-)Serialization
  - Transfer
- **Checkpoint/Restore** (Redundant Memory)
- **Hardware-supported Checkpoint/Restore**
  - Co-Processor (e.g. PB ES Cortex-M3)
  - FPGA-based (e.g. Enhanced MMU)
- **Hardware support (e.g. placement) of further migration related components** (Runtime Monitoring, Analysis, Planning)
- **Flexible thread management** (intelligent synchronisation vs. idle mode)
Conclusion

• Motivation: Future CPS
• Dynamic Reconfiguration: Migration of software-based components
• Relevant OS research areas
  • Mixed-Criticality-aware Partitioning
  • Migration decision
  • Migration execution
• Development
  • Implementation: L4 Fiasco.OC/Genode OS Framework
  • COTS-Hardware (physical/virtual)
• Test Setup & Test Cases
  • Hybrid Simulator
  • Automotive & Robotics
Thank you!

https://www.os.in.tum.de/

Sebastian Eckl, M.Sc.
TUM – Technische Universität München
Informatik F13 (Operating Systems)
Boltzmannstr. 3
85748 Garching
T: +49 89 289 18791
E: sebastian.eckl@tum.de
www.os.in.tum.de