artist

The ArtistDesign
European Network of Excellence
on Embedded Systems Design

http://www.artist-embedded.org/

Showcase of the Main Results

DATE Conference, March 15th, 2012

SEVENTH FRAMEWORK PROGRAMME

ArtistDesign European NoE: Showcase of the Main Results

DATE Conference, March 15th, 2012

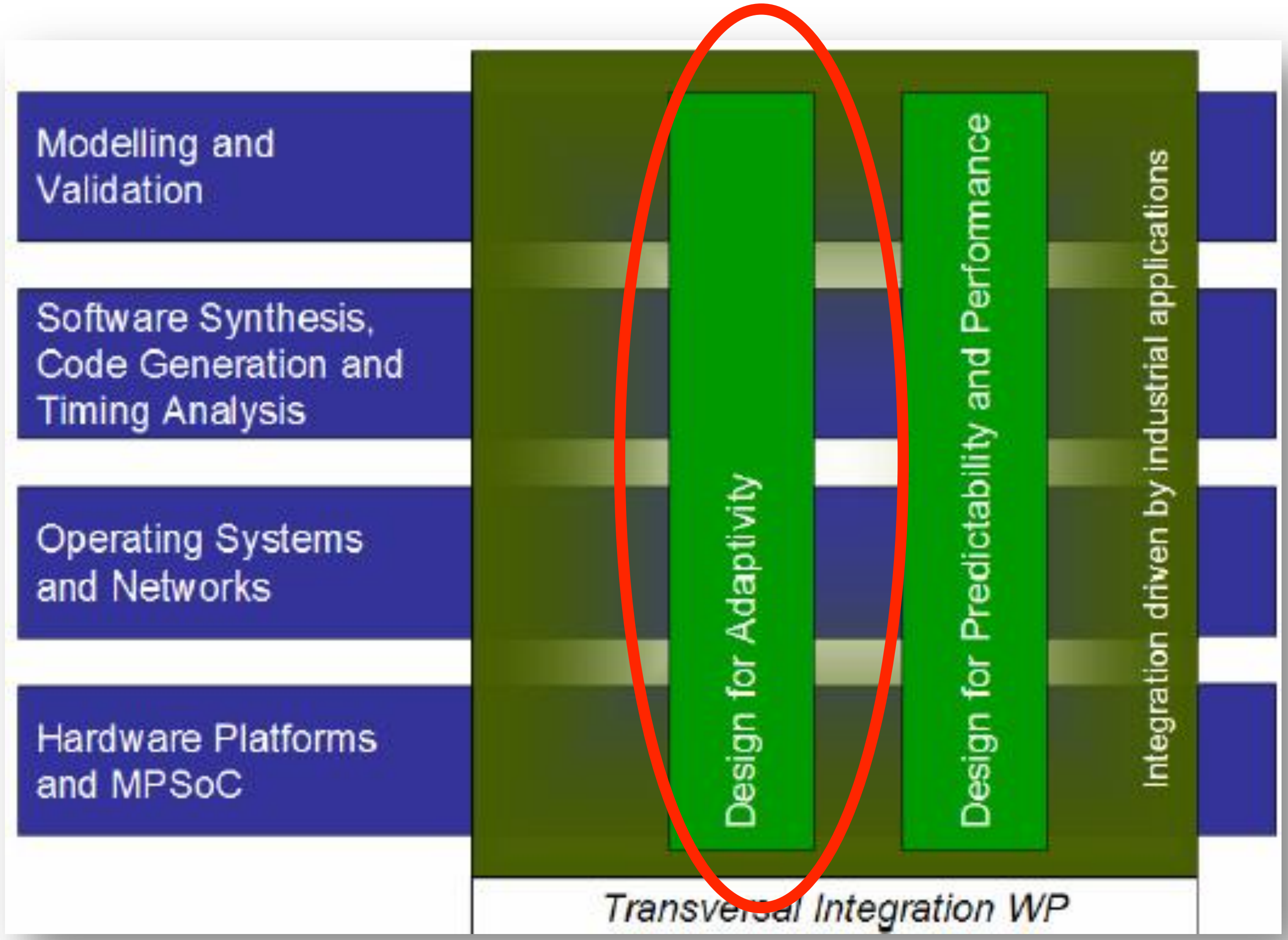
Achievements and Perspectives :

Design for Adaptivity

Karl-Erik Årzén

Lund University

Transversal Activity


Modelling and Validation
Software Synthesis, Code Generation and Timing Analysis
Operating Systems and Networks
Hardware Platforms and MPSoC
Design for Adaptivity
Design for Predictability and Performance
Integration driven by industrial applications
Transversal Integration WP

- Around half the size of a thematic cluster

Partners

Core Partners:

- Karl-Erik Årzén (ULUND)
- Gerhard Fohler (TUKL)
- Giorgio Buttazzo (SSSA)
- Luis Almeida (UPorto)
- Luca Benini (UBologna)
- Chantal Ykman-Couvreur (IMEC)
- Eduardo Tovar (IP Porto)
- Björn Lisper (MdH)
- Alan Burns (York)
- Lothar Thiele (ETH-Z)
- Hamid Brahim (CEA)

- Axel Jantsch & Martin Törngren (KTH)
- Jan Madsen (TU Denmark)
- Rolf Ernst (TUBraunschweig)
- Joseph Sifakis (VERIMAG)

Affiliated Partners:

- Alejandro Alonso (UPM)
- Lucia Lo Bello (UCatania)
- Pau Martí (UPC)
- Johan Eker (Ericsson)
- Liesbeth Steffens (NXP)

- Most partners from the OS and Networks cluster
- Hence, most focus on software-based approaches to adaptation

Contents

Definitions

*“An embedded system is **adaptive** if it is able to adjust its internal strategies to meet its objectives”*

*“An embedded system is **robust** if it meet its objectives under changing conditions without modifying its internal strategies”*

Why Adaptivity?

- System complexity increases
- Complete information about all use cases and their resource requirements is often not available at design-time
- Green computing → power consumtion constraints increasingly important
- Increased hardware density → thermal constraints increasingly important
- Hardware platforms increasingly complex → increasing difficulties in providing good off-line estimates of resource consumption
- Hardware variability increasing
- Increased requirements on dependability
- Hardware increasingly often allows adaptivity

Some Examples of Adaptivity

Example 1: A multi-mode embedded system where the resource requirements for all the tasks in all the modes are known at design time

- Use schedulability analysis to ensure that the deadlines are met in all modes and then use a mode-change protocol that ensures that all deadline also are met during the transition between the modes

Example 2: An embedded system with a constant set of hard-RT applications/tasks but where the WCET analysis possible on the selected hardware is too pessimistic and leads to too low resource utilization or where the age or process-induced variability is too large

- Measure the actual resource consumption and adjust, e.g. the task rates in order ensure that the schedulability condition is fulfilled

Examples of Adaptivity

Example 3: Open embedded systems where the number of applications and their characteristics change dynamically (e.g, smartphones)

– Measure resource consumption and decide how much resources that should be allocated to each application in order to maximize QoS/QoE while minimizing power consumpotion and avoiding thermal hotspots

Example 4: A distributed embedded system where we for dependability reasons must be able to ensure system functionality also in case of single-node failures

– Detect node failures and then adapt the task mapping and the schedules so that the the system performance is still acceptable

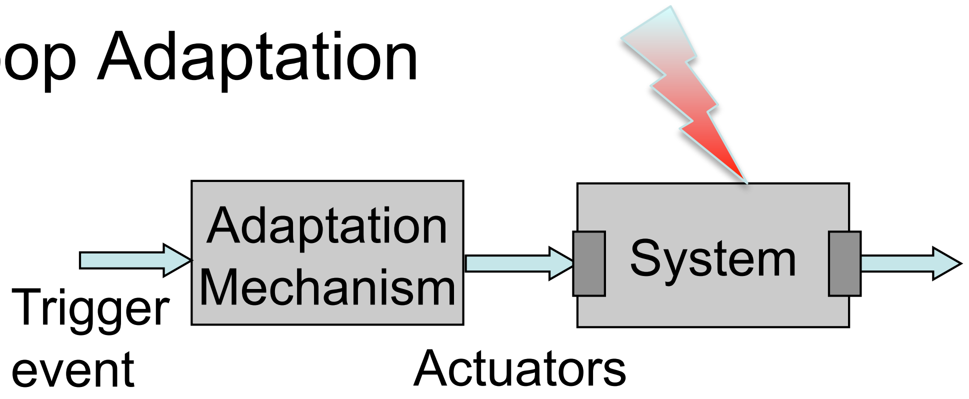
Examples of Adaptivity

Example 5: An FPGA-based system with multiple modes that is too large to fit in a single FPGA or where the power consumption will be too high

- Use run-time reconfiguration to change the FPGA function dynamically

Adaptation Mechanisms

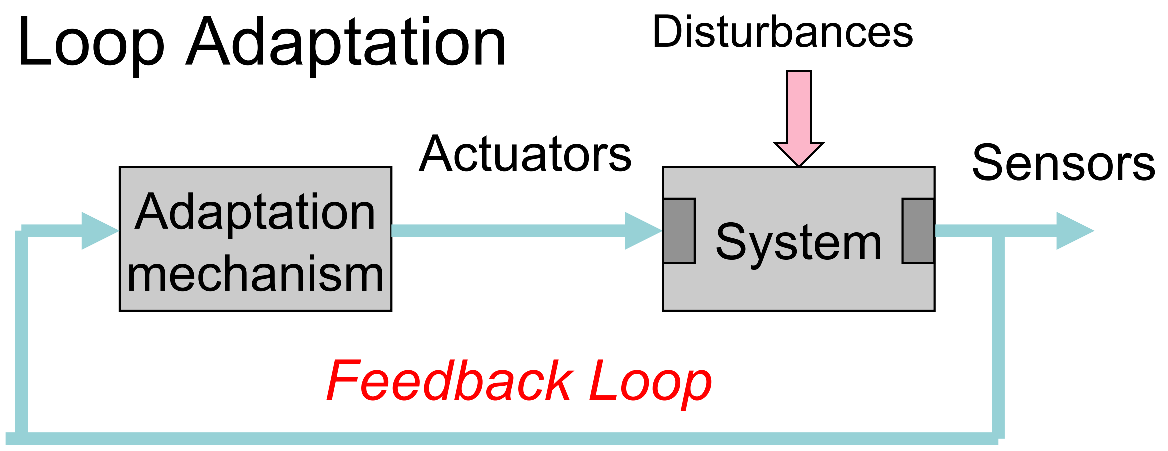
- Open Loop Adaptation



  - Feedforward
  - Assumes perfect information (model) of the system
  - Assumes that there are no external disturbances

Adaptation Mechanisms

- Closed Loop Adaptation



  - Feedback
  - Adaptation Mechanism == Controller
  - Requires sensors
  - May cause unstabilities

Adaptation Formulations

- Often formulated as an optimization-problem or as a control-problem
- Optimization Formulations:

maximize/minimize resource-consumption objective
s.t. perfomance constraint

or

maximize/minimize performance objective
s.t. resource consumption constraint

  - Performed off-line, online when some change has occurred or periodically, off-line+on-line, …
  - ILP, Bin-packing, MILP, QP, NLP (B&B, GA, CP …)
  - Centralized or distributed

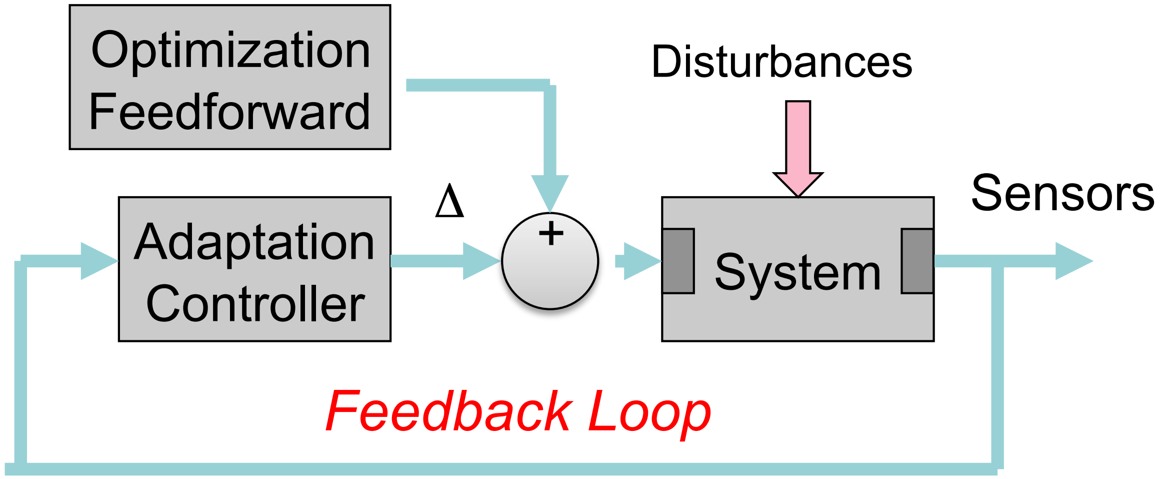
Adaptation Formulations

- Control Formulations:
  - System modelled as (linear) dynamic system
  - Classical linear control design techniques
    - PID
    - LQG
    - ….
  - Designed to obtain a stable closed loop system with desired dynamic performance

$$\boxed{u(t) = K(e(t) + \frac{1}{T_I}\int e(s)ds + T_D\frac{de(t)}{dt})}$$

Adaptation Formulations

- Combined Optimization and Control Formulations:
  - Model-Predictive Control (MPC)
    - Optimization problem solved each sample
    - Only the first control signal is used (receding horizon principle)
    - Optimization problem ban be solved off-line (explicit MPC / multiparametric programming) → piecewise affine mapping
  - Feedforward + feedback structures



Actuators

- Change the applications / threads
  - For example:
    - Accept or reject decision
    - Change the rates of periodic processes
    - Task shaping
    - Change between alternative versions (service/quality levels)
    - Anytime formulations
  - Often requires support from the applications
- Change the mapping of the application onto the execution platform
  - Priority
  - Schedule
  - Processor allocation

Actuators

- Change the execution platform
  - Number of processors (virtual or physical)
    - DPM techniques
  - Speed of processors
    - DVFS
    - Change the bandwidth of the VM or bandwidth server
  - Functionality (hardware-based systems)
    - Micro-code in soft-cores
    - FPGA netlist

Sensors

- What we can (or would we like to) measure?
  - Application performance
    - Obtained QoS
    - Throughput
    - Latency
  - OS / CPU level
    - CPU cycles / task
    - CPU utilization
    - Deadline miss ratio
  - Power and temperature
    - Power consumption for each unit
    - Temperature of each heat source (core, coprocessor, memory controller, ….)

Models

- It is unrealistic to assume sensors for everything
- Must be combined with realistic models that allow us to estimate entities which we cannot measure
- Dynamic calibration using sensor readings (Kalman filter / dynamic observers)
- Power models:
  - Dynamic and static power consumption
- Temperature models
  - Heat transfer between cores
  - Active cooling
  - Multi-tier 3D chips
- Interplay between power and temperature models
  - Temperature dependent leakage power
- Model parameters through system identification

Problems of Adaptivity

Adaptivity can introduce new problems:

- The adaptation mechanism itself consumes resources
- Harder to provide formal guarantees about the system
- Adds to the complexity
- May complicate the design process (modeling, V&V, …)
- Requires tuning
- Sensors and actuators are necessary
- Models are necessary
  - Of the system that we adapt
  - Of the adaptation mechanism itself

Contents

Selected Technical Highlights

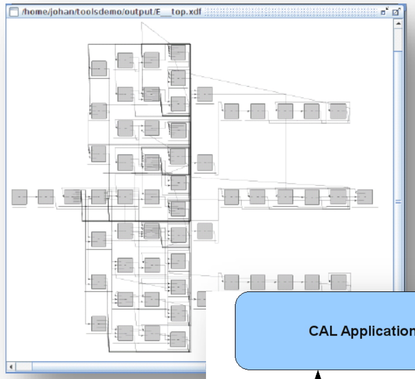
- Scheduling:
  - Analysis for dynamically changing task sets (UYork)
    - Maximise the utilisation of the available resources by adapting near optimal algorithms
  - Task allocation strategies that are robust to changes (UYork)
    - Minimizes the amount of change that has to be done to the system
  - Mode change protocols for bandwidth servers (SSSA, ETHZ)
    - Maintain schedulability and temporal isolation for CBS bandwidth servers when the server parameters change
  - A new allocation and scheduling approach for parallel tasks in a soft-real time media processing systems (UNIBO)
    - In the presence of variability on a multi-core platform.
    - Efficient online policy for meeting timing constraints with minimum energy.

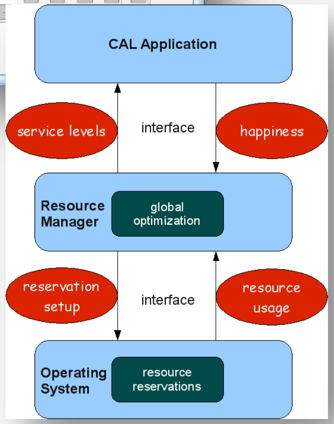
Selected Technical Highlights

- Memory:
  - Run-time adaptivity of the memory hierarchy within NoC architectures (UYork)
    - Real-time guarantees in dynamic systems
    - Reduce memory requirements and energy costs

Selected Technical Highlights

- Run-Time Resource Management:
  - ACTORS (SSSA, TUKL, ULUND, Ericsson)
    - Homogeneous multicore systems
    - Virtualization through CBS servers
    - Optimization (ILP) and control
    - New Linux scheduling class
    - Media processing and control applications implemented in the CAL dataflow language
    - Complete tool chain including design space exploration, simulation, model transformation, automatic code generation, profiling
    - X86, ARM11, ST-Ericsson dev board (Android)
    - Currently adding support for power management





Selected Technical Highlights

- Run-Time Resource Management:
  - Adaptive resource management for distributed multimedia systems (IMEC, BARCO)
    - Video clusters
    - BARCO reported a factor 3 reduction in average power consumption, a factor 5 reduction in hardware cost, and more than a factor 10 in system size as compared to their current solution.
  - Adaptive resource management (VERIMAG)
    - Multimedia applications with multiple quality levels that impacts the quality of service (QoS) and the execution times
    - Optimal quality levels computed online
    - Controller that is constantly adapting the chosen quality levels depending on the actual time and on a combination of average and worst-case estimates of the execution times

Selected Technical Highlights

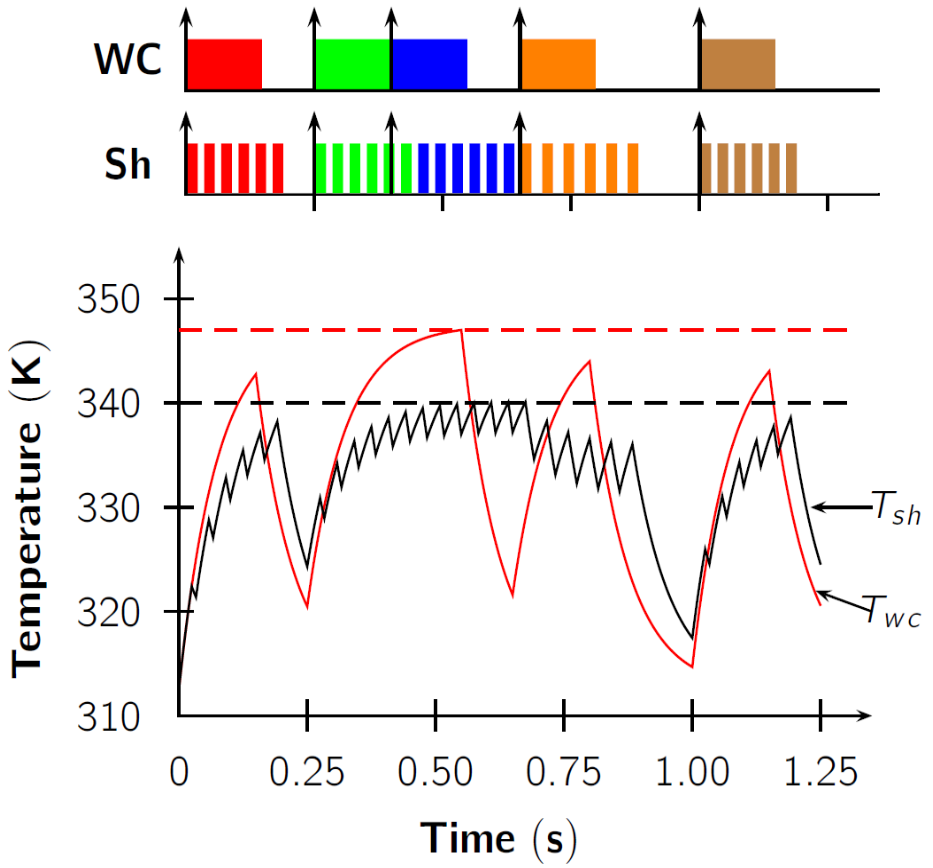
- Run-Time Resource Management:
  - Compile-time and run-time adaptivity for energy and variability (UYork)
    - Compiled code includes potential for OS to vary applications behaviour at run-time for achieving better performance
  - Adaptive Service Management (UPM)
    - Adaptation of service request handling behaviour to the specific requirements of the services
    - CPU contracts to ensure sufficient computation time (modifed Linux)
    - Quality compositions of services at run-time

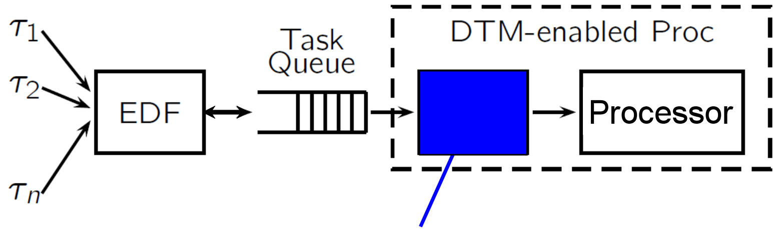
Selected Technical Highlights

- Run-Time Analysis:
  - Distributed run-time analysis of embedded systems (TUBraunschweig)
    - An existing analysis engine has been complemented by a framework that enables access control and runtime-optimization.
    - Extended with distributed algorithms that allow the usage of self-configuration services for self-protecting real-time systems.

Selected Technical Highlights

- Power and Thermal Management:
  - Task shaping to obtain thermal guarantees (ETHZ)
    - Just Sufficient Throttling (JUST)

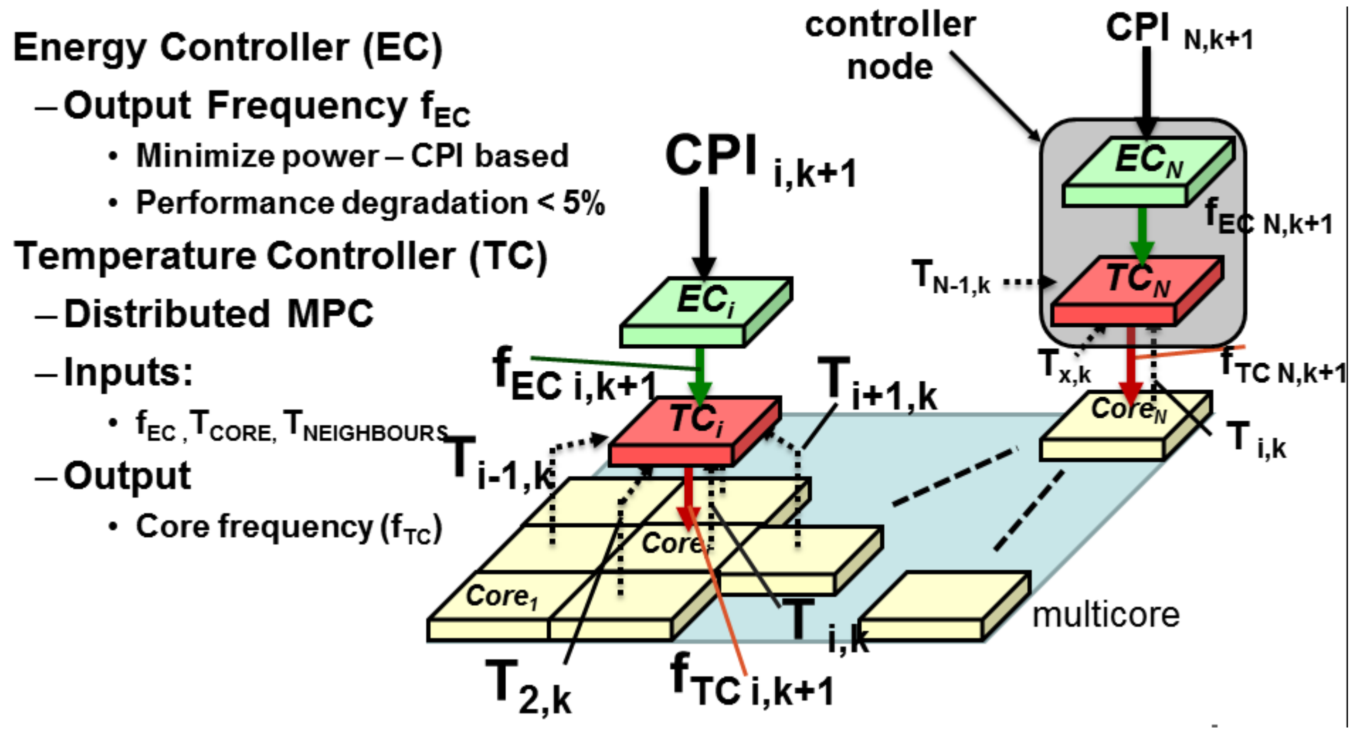




bounds the workload of the processor
tasks not delayed more than necessary

Selected Technical Highlights

- Power and Thermal Management:
  - Integrated energy and thermal control (UNIBO)
    - Cascaded control structure
    - Model-Predictive Control

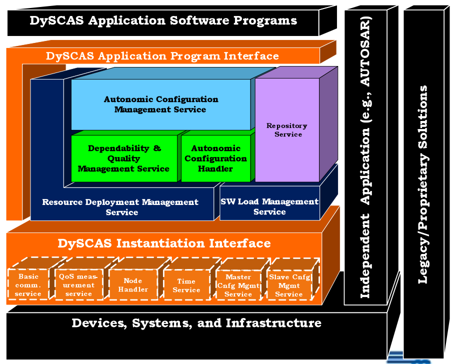


Energy Controller (EC)
- **Output Frequency f_{EC}**
  - **Minimize power – CPI based**
  - **Performance degradation < 5%**

Temperature Controller (TC)
- **Distributed MPC**
- **Inputs:**
  - **f_{EC} , T_{CORE}, $T_{NEIGHBOURS}$**
- **Output**
  - **Core frequency (f_{TC})**

Selected Technical Highlights

- Frameworks and Reference Architectures:
  - Adaptable Collaboration Framework (IPPorto)
    - Networked embedded systems
    - Allows constrained devices to collaborate with more powerful or less congested peers
    - Trade-off computation time and resource usage against quality
  - DySCAS - Dynamically Self-Configuring Automotive Systems (KTH, ++)
    - A reference middleware architecture for automotive embedded systems
    - Run-time support for the fusion of monitored context data, the resolution of conflicts and configuration variations, and the execution of dynamic adaptations (QoS changes and migrations)

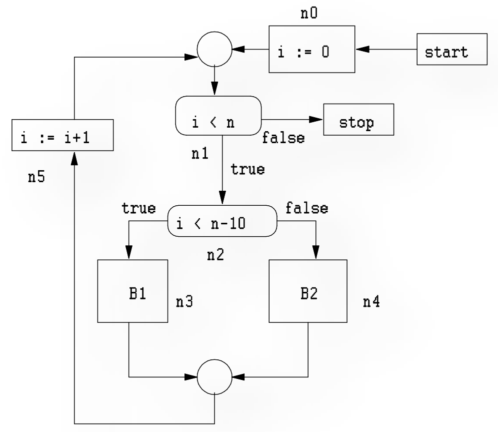

DySCAS Application Software Programs
DySCAS Application Program Interface
Autonomic Configuration Management Service
Repository Service
Dependability & Quality Management Service
Autonomic Configuration Handler
Resource Deployment Management Service
SW Load Management Service
DySCAS Instantiation Interface
Basic comm. service
QoS measurement service
Node Handler
Time Service
Master Cnfg Mgmt Service
Slave Cnfg Mgmt Service
Devices, Systems, and Infrastructure
Independent Application (e.g., AUTOSAR)
Legacy/Proprietary Solutions

Selected Technical Highlights

- Sensor Networks:
  - Adaptive energy management (ETHZ, UNIBO)
    - Sensor networks with solar cell-based energy harvesting
    - Adaptation of application parameters based on a prediction of future energy availability
    - Optimization problem solved using multiparametric programming
    - Optimal task scheduling using both time and energy constraints
- Adaptive Networking:
  - Communication channel adaptation (UPorto)
    - Virtual channels with adaptable bandwidth and latency through the Flexible Time-Triggered (FTT) approach over switched Ethernet
    - Adaptive TDMA that adapts its phase to escape interfering traffic

Selected Technical Highlights

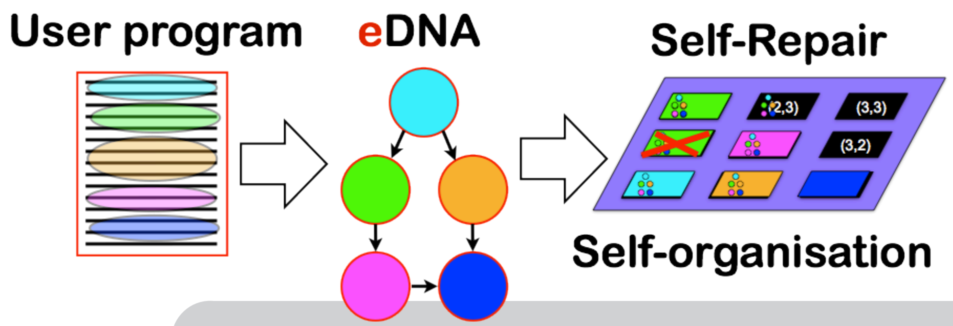
- Control Techniques:
  - Optimization of the timing parameters of real-time control tasks (SSSA)
  - New feedback scheduling techniques (UPC)
  - New event-based control techniques (ULUND, UPC)
- WCET Analysis:
  - Parametric WCET analysis (MDH)
    - Bounds as a function of input values
    - To be used in adaptive real-time systems where the task scheduling adapts to external factors
    - Analysis packaged in the SWEET tool



WCET(n) ≤ if $n \geq 11$: $190n - 530$
if $0 < n \leq 10$: $140n - 20$
otherwise: 20

Selected Technical Highlights

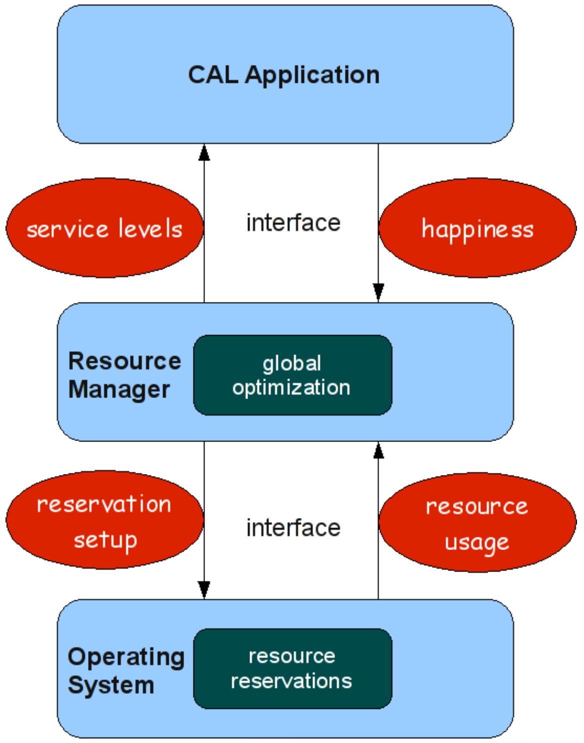
- Programmable Hardware:
  - eDNA architecture (DTU)
    - Ultra fault-tolerant FPGA
    - Multiple processors (cells) connected through NoC
    - Programmed via eDNA (electronic DNA), a behavioural spec of the user algorithm encoded in a binary format
    - Cells self-organize by translating the eDNA into tasks and maps them
    - In case of faults the self-organization algorithm is re-run
    - NASA JPL collaboration
    - Commercialized through spin-off company

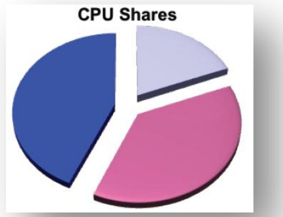


Contents

Feedback-Based Resource Management

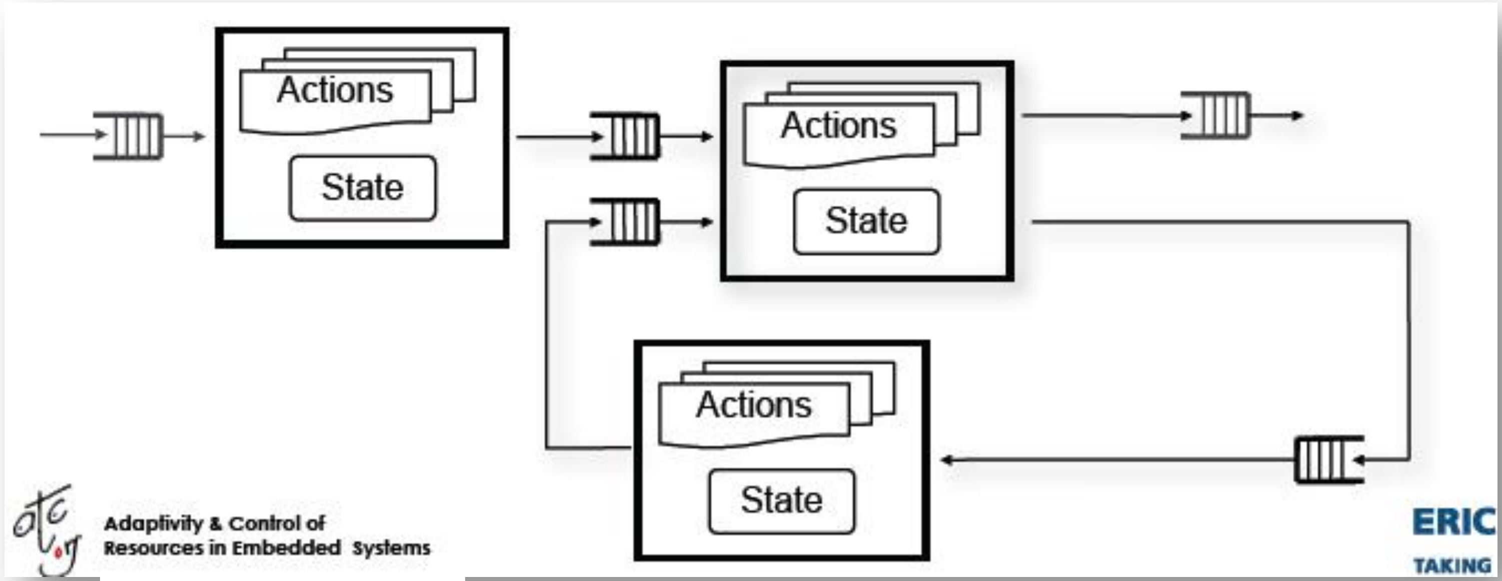
- ACTORS – Adaptivity and Control of Resources in Embedded Systems
  - Ericsson (coord), SSSA, TUKL, Lund, EPFL, Akatech, Evidence
- Levels:
  - Applications: CAL Dataflow Language
  - Resource Manager
  - Operating System: SCHED_EDF Linux scheduler (hard CBS) and Linux CFS scheduler
- Demonstrators
  - Media streaming on cellular phones, control, high-performance video
- Platforms: ARM 11 & x86 multicore with Linux > 2.6.26





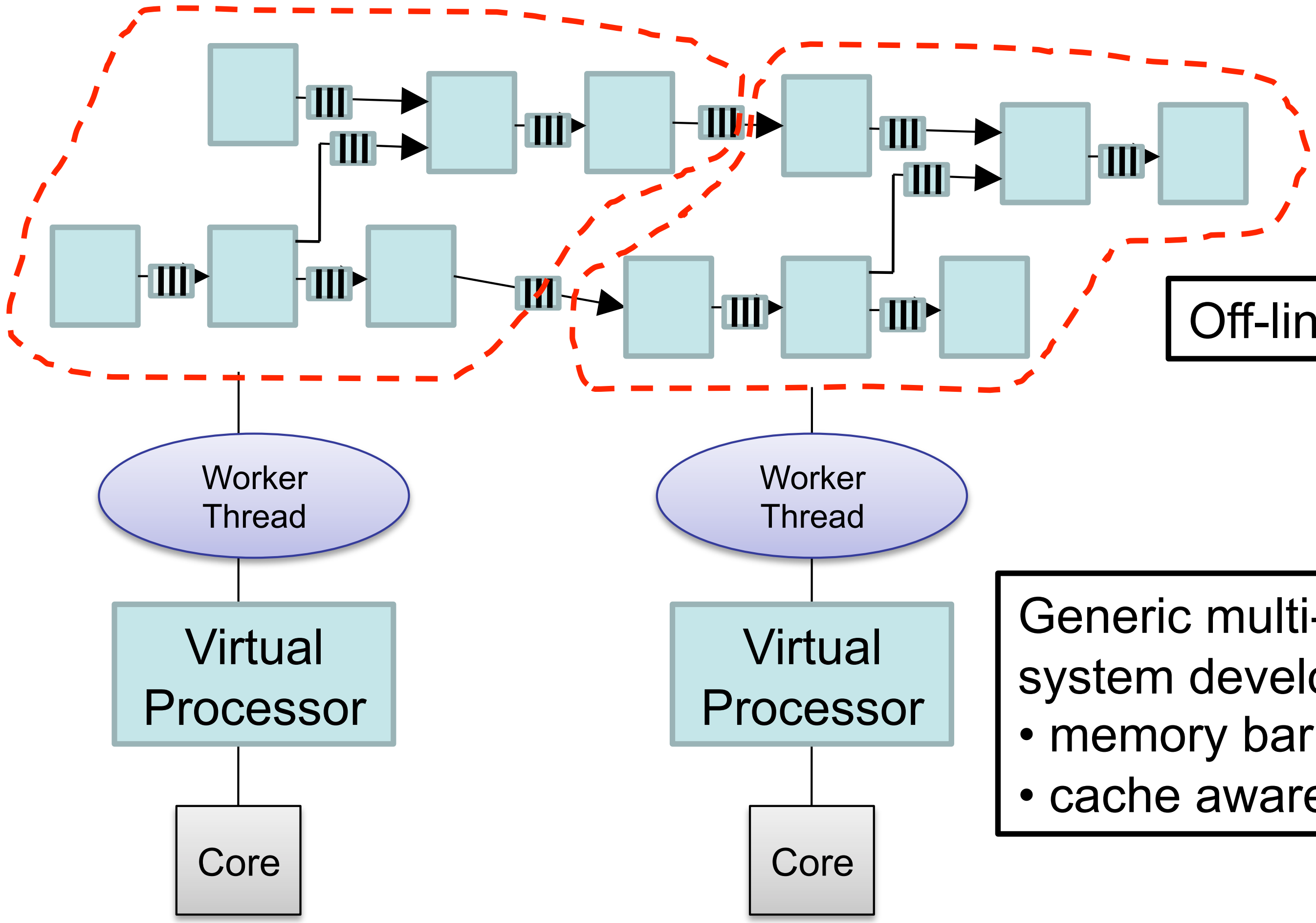
Dataflow Modeling

- Data flow programming with actors
- CAL Actor Language (UC Berkeley, Xilinx) http://opendf.org
  - Part of MPEG/RVC



Dataflow Execution

- Best-effort scheduling with dynamic processor allocation for dynamic CAL applications on multi-core platforms



Off-line Partitioning

Generic multi-core run-time system developed:
- memory barriers
- cache aware

Overview

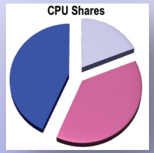
- CAL Dataflow Applications
- Legacy applications through wrapper

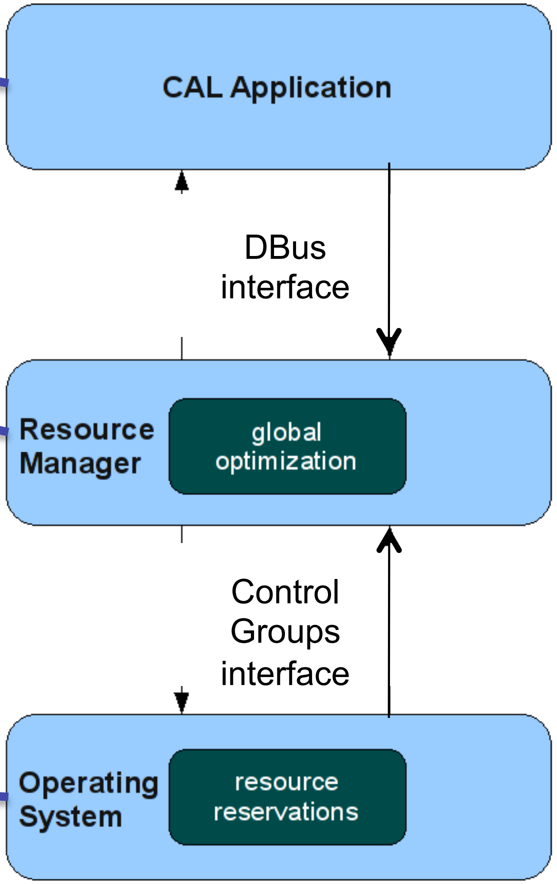
Resource Manager

- C++ framework
- DBus IPC to application
- Control groups API to scheduler

SCHED_EDF scheduler

- Partitioned multi-core scheduler
- Hard CBS Reservations





Static Information

From applications to RM at registration:

- Service Level Table

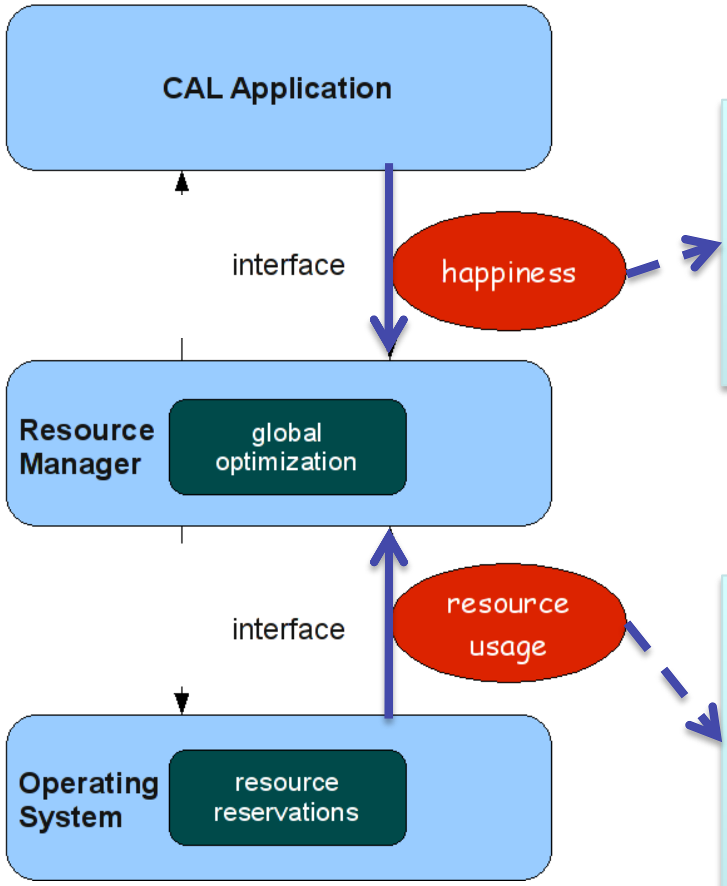
Service Level	QoS	BW Requirement	BW distribution	Timing Granularity
0	100	240	60-60-60-60	20 ms
1	75	180	45-45-45-45	20 ms
2	40	120	30-30-30-30	20 ms

- Thread IDs and how they should be grouped

From system administrator to RM at startup:

Appl.	Importance
Appl 1	10
Appl 2	20
Appl 3	100
Default	10

Dynamic Inputs



Happiness:

- boolean indicator of whether the QoS obtained correspond to what could be expected at the current service level

Used Budget (Bandwidth):

- average used budget

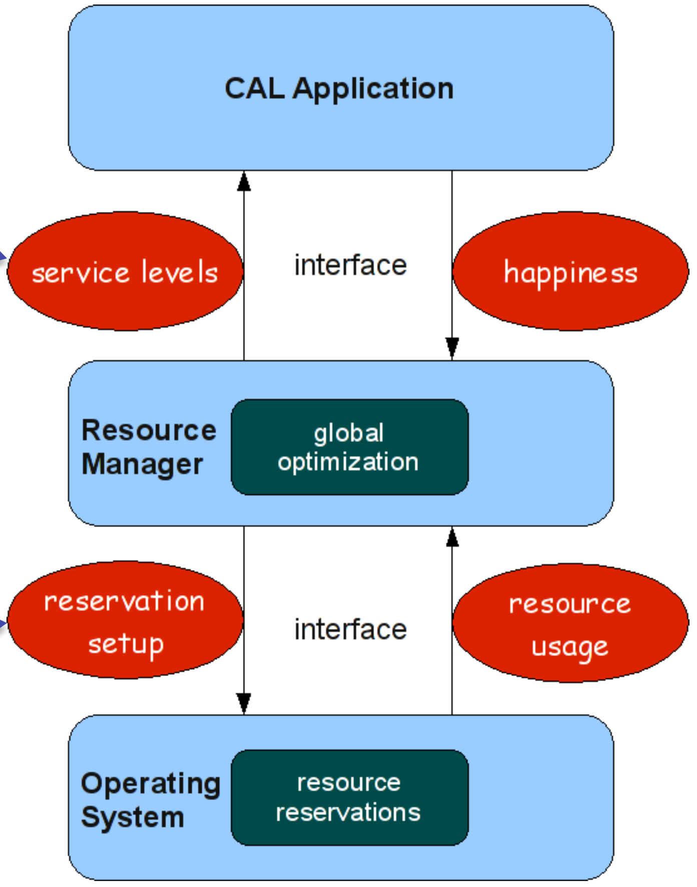
Exhaustion Percentage:

- percentage of server periods in which the budget was exhausted

Outputs

Current Service Level

Reservation Parameters:
- Budget
- Period
- Affinity



Resource Manager Tasks

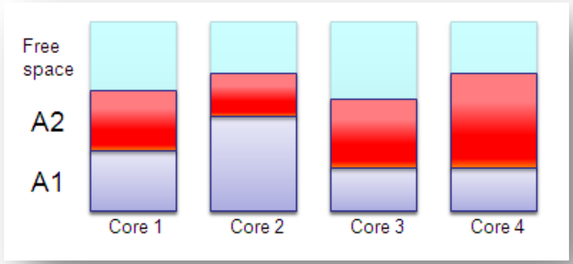
- Assign service levels
  - When applications register or unregister
- Mapping & bandwidth distribution
  - Map virtual cores to physical cores
- Bandwidth adaptation
  - Adjust the server budgets dynamically based on measured resource usage and obtained happiness

$$\max \sum_{i=1}^{n} w_i q_i x_i$$

$$\sum_{i=1}^{n} \alpha_i x_i \leq C$$

$$\forall i, \sum x_i = 1$$

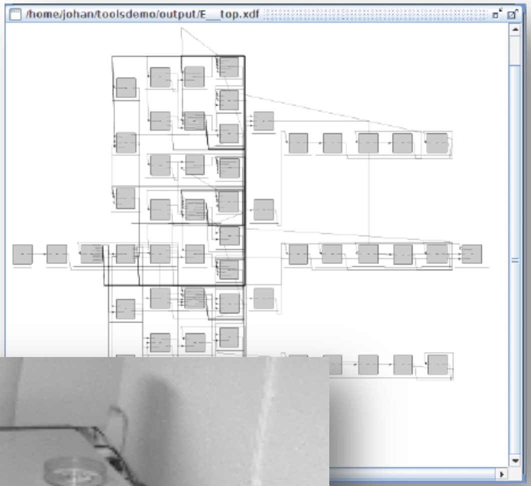
ILP Problem

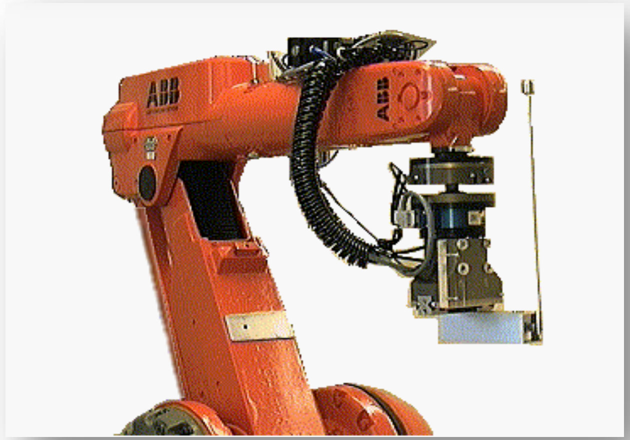


Bin-Packing Problem

Demonstrators

- Video Quality Adaptation
  - MPEG-2 and MPEG-4 frames
  - TUKL
- Control demonstrator
  - Industrial robot balancing inverted pendulum
  - Ball and Beam Processes
  - ULUND
- Image Processing
  - HW and/or SW mapping
  - EPFL





Video

Contents

- Motivation
- Selected Technical Highlights
- **Perspectives**

Insights Gained

Adaptivity in embedded systems covers a wide range of subjects.

Hence, to develop a common theoretical basis for adaptivity in embedded systems is extremely challenging.

The work performed within ArtistDesign can merely be considered as a starting point for this.

Insights Gained

In order to move adaptivity from the research community to industrial practice it is essential that adequate support for adaptivity is included in COTS software and hardware, including OS and middleware. This include

- *sensing and actuation mechanisms.*
- *models (thermal, battery, power, …) with correct parameters*
- *adaptivity API between applications and OS/middleware*

Insights Gained

There is a fundamental trade off between adaptivity and predictability. Hence, for applications with severe requirements on predictability, adaptive mechanisms are less suitable.

Furthermore, adaptivity makes formal verification more difficult

→ *So maybe it is not suitable for hard RT, time-critical systems?*

Or?

As soon as fault tolerance and reconfigurability becomes design requirements we have in essence adaptivity

Insights Gained

1) The adaptation mechanisms must be very resource efficient.

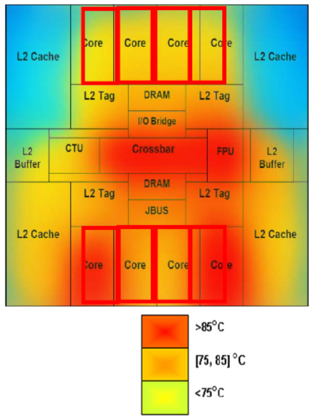
2) The requirements which they pose on the applications and the knowledge they require about the applications must be small.

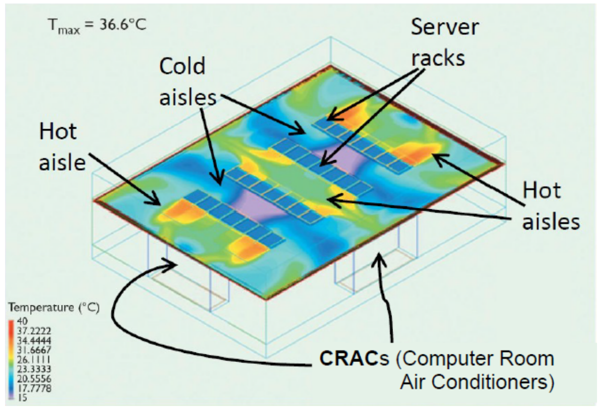
⇓

The adaptation mechanisms must be quite simple in order to be practically useful.

Insights Gained

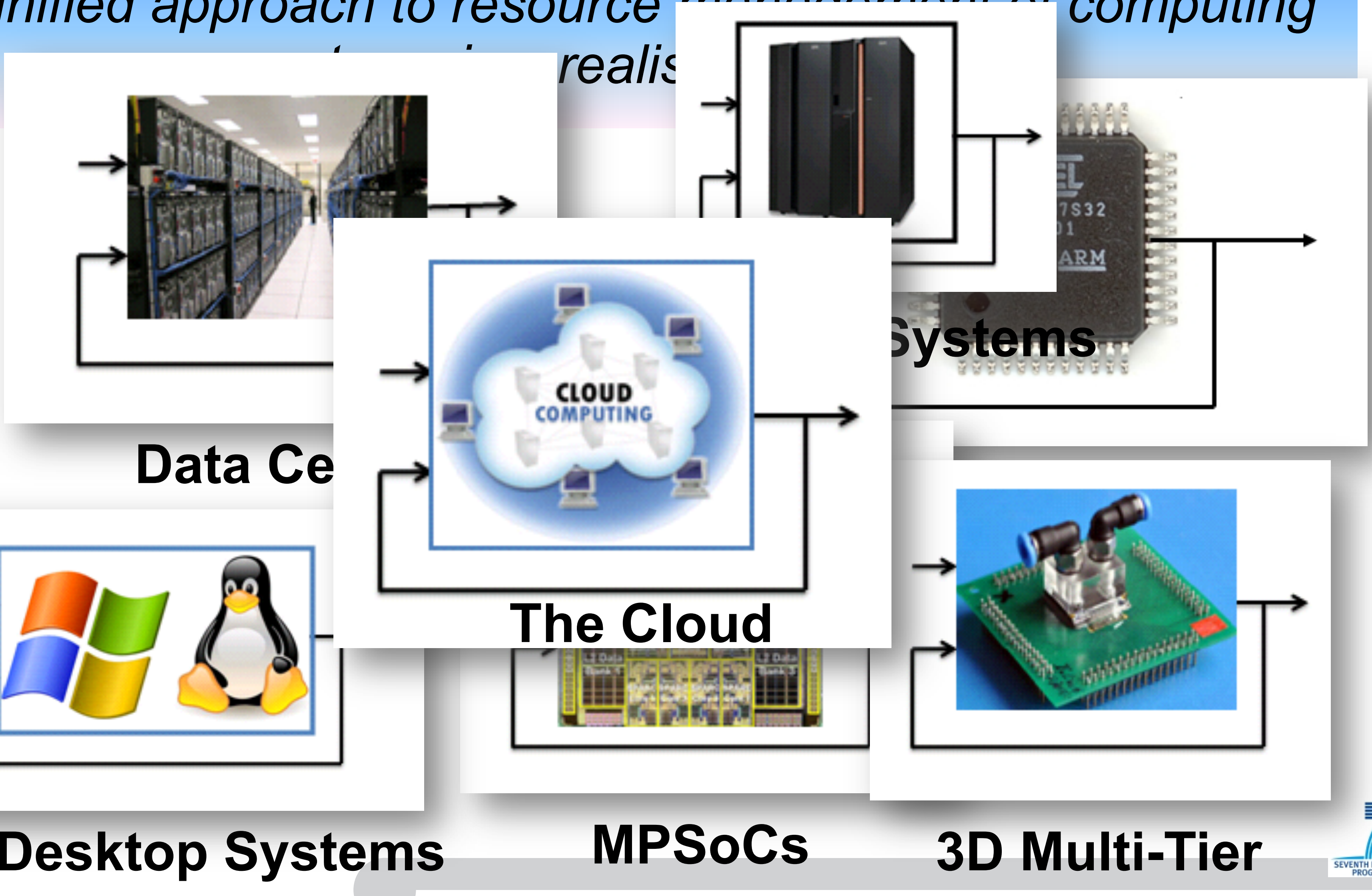
The thermal control, power control, and performance control needed in multi/many-core embedded systems have very strong relationships with the same problems in data centers.





Insights Gained

A unified approach to resource ~~management of~~ computing ... realis...



Questions?