Advanced Computer Architectures

A Safari Through the MPSoC RTM Jungle
Runtime Management was created to mediate the applications needs, user requirements or multiple simultaneously executing applications competing for platform resources
Multi-Processor System on Chip

Key Requirements:

- High Performance
- Low Power Consumption
- Easy Programmability
- Predictability
- Flexibility and Scalability

.... What is the end-user interested in?
**Architecture**

- **Sweet spot**
- You can find the RTM in the middle of Application layer and the platform services
- Quality manager or Quality of service interact with applications

Subfunctions: QoE & Operating point selection
Concept of Quality: introduction to Data Structure

Application has quality level \([q_{ij}] \rightarrow \) Levels associated with the Platform Resource Vector.

\[ R_{ij} = [r_{ij}^p, r_{ij}^m, r_{ij}^c] \]

- Processing
- memory
- communication

The third value is called implementation details, but this is not considered at this stage...

- Together this values are combined and forwarded to the subfunction of the QM.

- The aim is to maximize the total application value!
Quality Manager Sub-functions

By means, selecting an operating point 
(vij, rij, dij)
for every active application so that the total system utility is maximized, while the total amount of required resources does not exceed the available platform resources.

Session Utility Function is a relative value Vij for every qij in a certain moment.
**Taxonomy (Policies)**

- Application operating point selected
- Deciding the allocation of resources
- Resource management Algorithm
- Solution Space
Taxonomy of resource assignment algorithms (policies) for parallel and distributed systems

Mapping Speed vs Mapping Quality
**Run-Time Library**

Two functions:

- Provides Primitives called by Apps to abstract services provided by the hardware.
- Interface to system manager, enforce the decisions!

Primitives:

- Quality Management: renegotiation of selected quality level
- Data and communication management: Messages, tasks, Allocate memory, memory hierarchy
- Task management: Interaction, Create & Destroy tasks. Real link RM - QM
**HOW: Distributed Configurations**

**Master-Slave**
- One RTM on Master.
- Pro: Efficient for intensive jobs
- Con: Master is the Bottleneck

**Separate Supervisor**
- Every Processor has a dedicated RTM
- Existing collaboration structure
- Pro: No bottleneck
- Con: Duplication of data & memory penalty

**Symmetric**
- Single RTM executed by all Processors
- Pro: Most flexible configuration
- Con: Difficult to implement and manage
Adaptation allows to achieve requirements and constraints in an efficient way by exploiting application knowledge gathered at run-time.

**Hardware**
- Faster
- Better response time
- Increased real time Behaviour
- Mitigation of Core complexity
- Avoid overhead on App processor
- Energy efficiency
- Real time behaviour
- Less Cache Space
- Centralized
- Non Adaptive

**Software**
- Better Application Scalability
- Adaptive
- Distributed
Adaption: provides fast response time for interactive applications, high throughput for batch applications and an amount of fairness between applications

Run Time:

3 types of negotiations.
- With adaptive applications
- Configuring RTM policies (Tuning)
- Application takes over RTM responsibilities

Design time Adaptation
Multiprocessor Run-Time Management Examples

Texas Instruments

- Run-time manager is a software implementation of a Master-Slave configuration.
- The resource manager is responsible for selecting and allocating the slave DSP, for task creation and for setting up the communication structures, for starting and stopping the tasks and, finally, for deallocating the resources.
- Application developers can build modern multi-threaded applications in an easy way.

ARM MPCore & Linux SMP

- Homogeneous embedded multiprocessor platform that relies on a general purpose operating system.
- The Operative system hides the different processing engines
- Enables an easy speed-up of the applications:

K42.

- an IBM research run-time manager for 64-bit cache-coherent multiprocessor platforms.
- Focuses on high performance, platform scalability and application adaptivity.
- Although K42 implements a Symmetric configuration on the surface, every run-time manager resource object and associated data structures can be distributed in an efficient way over the multiprocessor in order to exploit the use of local memory and to avoid global data structures, global data locks and global management policies.
- Just like a Separate Supervisor approach, this approach provides near linear scalability.
## RTM Architectures Overview

<table>
<thead>
<tr>
<th>RTM Implement.</th>
<th>Distribution</th>
<th>Adaptivity</th>
<th>HW/SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI OMAP</td>
<td>Master-Slave</td>
<td>None</td>
<td>SW</td>
</tr>
<tr>
<td>Linux</td>
<td>Symmetric</td>
<td>None</td>
<td>SW</td>
</tr>
<tr>
<td>ThreadX</td>
<td>Symmetric</td>
<td>Design-Time</td>
<td>SW</td>
</tr>
<tr>
<td>Quadros RTXC</td>
<td>Separate Supervisor</td>
<td>Design-Time</td>
<td>SW</td>
</tr>
<tr>
<td>ST MultiFlex</td>
<td>Master-Slave and symmetric</td>
<td>None</td>
<td>HW RTLib</td>
</tr>
<tr>
<td>RealFAST RTU</td>
<td>Master-Slave</td>
<td>None</td>
<td>HW RTLib</td>
</tr>
<tr>
<td>Eclipse</td>
<td>Master-Slave</td>
<td>None</td>
<td>HW RTLib</td>
</tr>
<tr>
<td>Odyssey</td>
<td>Symmetric</td>
<td>Run-Time, type 1</td>
<td>SW</td>
</tr>
<tr>
<td>Large-scale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cosy</td>
<td>Separate Supervisor</td>
<td>None</td>
<td>SW</td>
</tr>
<tr>
<td>AsyMOS</td>
<td>Master-Slave</td>
<td>Run-Time, type 3</td>
<td>SW</td>
</tr>
<tr>
<td>K42</td>
<td>Symmetric with separate supervisor</td>
<td>Run-Time, type 2</td>
<td>SW</td>
</tr>
<tr>
<td>MIT Exokernel</td>
<td>Symmetric with separate supervisor</td>
<td>Run-Time, type 3</td>
<td>SW</td>
</tr>
<tr>
<td>Intel McRT</td>
<td>Master-Slave with separate supervisor</td>
<td>Design-Time and</td>
<td>SW with HW</td>
</tr>
<tr>
<td></td>
<td>properties</td>
<td>Run-Time, type 2</td>
<td>RTLlib support</td>
</tr>
</tbody>
</table>
Upcoming Research Challenges

- Never ending story
- Predictability Issues
- RTM monitoring and management for Platform HW reability

Conclusions...

- Moderate distribution of RTM over platform resources
- More platform services to support RTM
- More configurability towards applications
DISCLAIMER: Open Office

This Presentation was done with Open office that is famous for two main reasons:

- It's Free!
- It Sucks!