A Mobile Agent-based System for Dynamic Task Allocation in Clusters of Embedded Smart Cameras

Michael Bramberger\textsuperscript{1,2}, Bernhard Rinner\textsuperscript{1}, and Helmut Schwabach\textsuperscript{2}

1 Institute for Technical Informatics Graz University of Technology AUSTRIA
{bramberger, rinner}@iti.tugraz.at

2 Video & Safety Technology ARC Seibersdorf Research AUSTRIA
helmut.schwabach@arcs.ac.at

May 20, 2005 Hamburg, Germany

http://www.iti.tugraz.at/smartcam

Overview

- Introduction
- The Smart Camera
- Surveillance System Architecture
- Task Allocation
- Implementation & Experiments
- Outlook

Surveillance Systems

- 1\textsuperscript{st} and 2\textsuperscript{nd} generation
  - Primarily analog frontends
  - Backend systems are digital
- 3\textsuperscript{rd} generation
  - All-digital systems
- 3\textsuperscript{+} generation
  - Smart cameras
  - Surveillance tasks run on-site on smart cameras
    - Video compression
    - Traffic statistics
    - Accident detection
    - Wrong-way drivers
    - Stationary vehicles (tunnels)
    - Tracking

The Smart Camera - Hardware

- Heterogeneous multi-processor system
  - CMOS image-sensor
  - TI TMS320C64x DSPs
  - Intel IXP4xx network processor
- Scalable HW design
  - Up to 10 DSPs
- Interfaces
  - 100Mbit Ethernet, USB, Serial, GPRS, WLAN
The Smart Camera - Software

- Two-Fold Software Framework
  - DSP-Framework
  - SmartCam-Framework on Network Processor
- Provides services
  - Data Management
  - Performance monitoring
  - Synchronization
  - Message-based communication DSP→DSP / DSP→XScale
- Dynamic loading of modules (binaries) on DSPs

Surveillance System Architecture

- Improved surveillance due to smart cameras
  - Decentralized, distributed surveillance
  - Autonomous allocation & reconfiguration of tasks
- Functional groups of smart cameras
  - Limited performance and resources
    - Real-Time Constraints
  - Distribute sets of surveillance tasks to groups of smart cameras
    - “Surveillance Clusters”

Task Allocation 1

- Determine optimal allocation of tasks to smart cameras
  - According to required resources – primarily of the DSPs
    - CPU, MEM (on-/off-chip), DMA (channels, IRQs), transferred data
  - Overloading of a resource yields in violation of real-time constraints
- Reallocation is required **online** due to
  - Changed resource requirements of tasks (raised by software)
  - Changed resource availability of a camera (raised by hardware)
  - Added/removed cameras or tasks
  - Tracking tasks migrates between hosts

Task Allocation 2

- Is a constraint-satisfaction-problem (CSP)
  - Variables → Tasks \( \{T_1, \ldots, T_n\} \)
  - Domain → Cameras \( \{H_1, \ldots, H_m\} \)
  - Constraints → Allocation of all tasks
    - No camera must be overloaded
  - Complexity: \( O(m^n) \)
  - Resources considered in constraints
    - CPU, MEM (on-/off-chip), DMA (channels,IRQ), and transferred data
  - State-of-the-art approaches
    - provide one solution
    - require a central host
Task Allocation

- Is a constraint-satisfaction-problem (CSP)
  - Variables, Domain, and Constraints
  - Standard approaches provide one solution, require a central host

- Splitting the domain → Create a distributed CSP (DCSP)
  - Determine partial allocation (solutions of sub-CSP) in parallel
  - Prune set of partial allocations using a cost function
  - Merge partial allocations
  - Exploit distributed architecture

Determine Partial Allocations

- Determine Partial Allocations (solutions of sub-CSP) in parallel
- Prune set of partial allocations using a cost function
- Merge partial allocations
- Exploit distributed architecture

Determine Final Allocations

- Final allocations based on partial allocations
- Resource allocation and task mapping
- Optimization and resource allocation

http://www.iti.tugraz.at/smartcam
Cost Function

- **Goal**
  - Reduce number of allocations (early pruning)
  - Choose optimal allocation

- **Five cost classes**
  - Resource Costs → Data-Transfer Costs
  - Migration Costs → Affinity Costs
  - Quality-of-Service Costs

- Costs are based on availability of resource

Integrating the Cost Function

- **Remove allocations**
  - Calculate Cost for each task on every camera
  - Remove allocations which exceed an adaptive threshold

- **Find optimal solution**
  - Determine the cost of each feasible allocation
  - Choose the allocation which provides the lowest cost value

Integrating the Cost Function 1

- **Remove allocations**
  - Calculate Cost for each task on every camera
  - Remove allocations which exceed an adaptive threshold

- **Find optimal solution**
  - Determine the cost of each feasible allocation
  - Choose the allocation which provides the lowest cost value

Integrating the Cost Function 2

- **Cost integration reduces number of allocations**
  - Threshold is set according to lowest cost
    - A minimum number of feasible allocations has to be accepted

- **Reallocation Scenarios**
  - Increased system load
    - Prune set of feasible allocations
    - Fast results required
  - Reduced system load
    - Determine new set of feasible allocations
    - Timing to achieve results is relaxed
Implementation — System Setup

- Operational environment
  - Network processor running Linux
  - DSPs running DSP/BIOS (TI)
  - SmartCam/DSP-Framework

- Mobile agent-based implementation
  - Diet-Agents agent system
  - JamVM as Java virtual machine
  - Tight interface to DSPs

Implementation — Agent System

- DSP-Agencies
  - Enhanced agencies to integrate DSPs
  - Including
    - DSP Interaction Agent — DIA
    - Cluster Information Agent — CIA
    - System Agent — SA
    - Worker Agents — WA

- DSP-Agents
  - DSP module (binary)
  - Several QoS levels
  - Store intermediate data

Prototype

- Intel IXDP425 Baseboard
  - Intel IXP425 Network Processor @ 533 MHz
  - 256 MB SDRAM, 2x 100BaseT Ethernet, PCI

- ATEME IEKC64 (2x)
  - Texas Instruments TMS320C6416 @ 600 MHz
  - 264 MB SDRAM
  - National Semiconductor LM9618 CMOS image sensor

- WaveCom GSM/GPRS module
  - Supports Class 10 data transfers
  - Backup network connection
Prototype

Experimental Results

- Experiments conducted on
  - a prototype of the smart camera,
  - two 1 GHz Pentium-III PCs

<table>
<thead>
<tr>
<th>Case</th>
<th>Executed by</th>
<th>PC JDK 1.4</th>
<th>PC JamVM</th>
<th>SmartCam JamVM</th>
<th>SmartCam Native/C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a  Partial Allocations (6 Ag.)</td>
<td>20 ms</td>
<td>14 ms</td>
<td>79 ms</td>
<td>9 ms</td>
<td>13 ms</td>
</tr>
<tr>
<td>1b  Partial Allocations (3 Ag.)</td>
<td>14 ms</td>
<td>7 ms</td>
<td>55 ms</td>
<td>9 ms</td>
<td>13 ms</td>
</tr>
<tr>
<td>2a  Merge Solutions (6 Ag.)</td>
<td>766 ms</td>
<td>4.852 ms</td>
<td>21.363 ms</td>
<td>2.360 ms</td>
<td></td>
</tr>
<tr>
<td>2b  Merge Solutions (3 Ag.)</td>
<td>9 ms</td>
<td>5 ms</td>
<td>31 ms</td>
<td>4 ms</td>
<td></td>
</tr>
<tr>
<td>3a  Overall allocation determination (6 Ag.)</td>
<td>786 ms</td>
<td>4.866 ms</td>
<td>21.442 ms</td>
<td>2.373 ms</td>
<td></td>
</tr>
<tr>
<td>3b  Overall allocation determination (3 Ag.)</td>
<td>23 ms</td>
<td>12 ms</td>
<td>86 ms</td>
<td>13 ms</td>
<td></td>
</tr>
<tr>
<td>4   Reusing allocations (6 Ag.)</td>
<td>14 ms</td>
<td>32 ms</td>
<td>172 ms</td>
<td>26 ms</td>
<td></td>
</tr>
</tbody>
</table>

Outlook

- Evaluate more complex scenarios
- Real-world tests
- Integrate learning agents to optimize the reallocation behavior
- Performance optimizations