MULTICORE EXECUTION OF DYNAMIC DATAFLOW PROGRAMS ON THE DISTRIBUTED APPLICATION LAYER

Jani Boutellier and Amanullah Ghazi
OVERVIEW

• A design flow for mapping applications written in a *dataflow language* to multicore and GPU platforms is proposed

• The proposed design flow is based on two major software frameworks

• Experiments are performed on two application use cases: parallelized video motion detection and predistortion filtering
DATAFLOW (1/2): INTRODUCTION

• The dataflow Model of Computation fits well to execution of concurrent applications on parallel platforms
• The dataflow concept has been refined into variants that balance between analyzability and expressiveness
• In all dataflow variants, programs are expressed as networks of actors that are interconnected by FIFO queues
DATAFLOW (2/2): RVC-CAL

- RVC-CAL is an ISO-standardized dataflow language
- The model of computation under RVC-CAL is Dataflow Process Networks (DPN) by Lee et al (1995)
- DPN maximizes expressiveness with the cost of reduced analyzability
PROPOSED DESIGN FLOW (1/4): DAL

• The target of our design flow is to map dataflow programs to multicores and GPUs
• Distributed Application Layer (DAL) by ETH Zürich is a framework for executing concurrent applications on platforms that consist of multicore processors (e.g. Intel Xeon Phi) and GPUs
• The Model of Computation assumed by DAL is Kahn process networks
PROPOSED DESIGN FLOW (2/4)

RVC-CAL Dataflow Program

Open RVC-CAL Compiler

DAL Backend

processes network mapping

Distributed Application Layer

Multicore Platform
PROPOSED DESIGN FLOW (3/4): TRANSLATING ACTORS TO PROCESSES

RVC-CAL actor

\[ \text{initialize} \]

\[ a \rightarrow c \rightarrow b \]

\[ \text{fire} \]

\[ \text{init} \]

\[ \text{finish} \]

DAL process
PROPOSED DESIGN FLOW (4/4): TARGET PLATFORM CODE GENERATION

- Once the actors of the dataflow program have been transformed to a DAL KPN, DAL can execute the program on multicores and GPUs
- However, DAL has several backends for targets such as Intel Xeon Phi, GPU (OpenCL), Intel SCC and Linux-based generic multicores
- The target platform is described in an XML file, where the core types and parameters are given, as well as the interconnect
EXPERIMENTS (1/2)

- Two applications were executed on two multicore platforms

<table>
<thead>
<tr>
<th>Platforms used for experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Xeon</td>
</tr>
<tr>
<td>Opteron</td>
</tr>
</tbody>
</table>
EXPERIMENTS (2/2)

• The performance provided by the proposed design flow was compared against the multicore code generated directly by Orcc
• The number of cores used varied between 1 and 16
• An experiment was also conducted to see the effect of FIFO size to speedup
RESULTS (1/5)

- Speedup of motion detection on the Xeon platform

![Graph showing speedup of motion detection on the Xeon platform]
RESULTS (2/5)

- Speedup of motion detection on the Opteron platform
RESULTS (3/5)

• Speedup of predistortion on the Xeon platform

![Graph showing speedup of predistortion with cores on the x-axis and megasamples/s on the y-axis. The graph shows a trend indicating increased speedup as the number of cores increases.]
RESULTS (4/5)

• Speedup of predistortion on the Opteron platform
RESULTS (5/5)

Single-core throughput of predistortion. X axis is $\log_2$ of FIFO size. $\triangle = \text{Orcc native on Xeon}; \diamond = \text{Orcc native on Opteron}; \circ = \text{proposed on Opteron}; \square = \text{proposed on Xeon}.$
CONCLUSIONS

• With the platforms used in the experiments the proposed platform clearly yields a higher speedup when compared to the regular multicore generated by Orcc.
• However, the final experiment showed that with small FIFO sizes Orcc clearly outperforms the proposed approach.
• Both the reference approach and the proposed one relied on Linux inter-process communication, yet the DAL framework relies on mutexes and the reference approach on semaphores.
• An interesting direction for future work would be to discover if the advantages of both approaches could be combined.
Thank you for your attention!

Questions?