Application Autotuning to Support Runtime Adaptivity in Multicore Architectures

Davide Gadioli, Gianluca Palermo, Cristina Silvano
Politecnico di Milano
cristina.silvano@polimi.it

SAMOS 2015
Motivations

Autotuning is a key issue for enabling applications running on many-core architectures to operate close to optimal efficiency in the face of:

- Changing conditions by adjusting their behavior to their operating environments, usage contexts and resource availability
- Meeting the requirements on performance, power consumption and Quality-of-Service
Application Autotuning

- In the context of autonomic computing, application autotuning enables self-optimization capabilities, based on the Monitor-Analyze-Plan-Execute (MAPE) feedback loop as defined in [1]:

Run-Time Resource Management

Introduction

- This talk introduces ARGO, a light-weight application auto-tuning framework for multi/many-core platforms in an adaptive multi-application environment.

- The autotuning framework should manage at runtime the assignment of system resources to the active concurrent applications (request-level parallelism)
ARGO Application Autotuning

- **ARGO autotuning approach** exploits the concepts of:
  - **Orthogonality** between application autotuning and runtime management of system resources to support multiple adaptive applications.
  - Combination of **design-time** and **run-time** techniques to create an effective way of “self-aware” computing with **limited runtime overhead**.
ARGO Application Autotuning

- **Key idea** is that most of the applications are configurable in terms of a set of parameters by using **run-time knobs** to trade-off Quality of Results and Throughput.
design time

run time

QoR

Throughput

8
Why Run-Time Knobs & Autotuning?

- In some applications, internal knobs can be used at run-time to trade-off quality of results and throughput.
Why Run-Time Knobs & Autotuning?

- In some applications internal knobs can be used at run-time to trade-off quality of results and throughput.

- Video Frame Rate
- Video Resolution

Autonomous Video-surveillance System
Why Run-Time Knobs & Autotuning?

- In some applications internal knobs can be used at run-time to trade-off quality of results and throughput
Why Run-Time Knobs & Autotuning?

- In some applications internal knobs can be used at run-time to trade-off quality of results and throughput
Why Run-Time Knobs & Autotuning?

- In some applications internal knobs can be used at run-time to trade-off quality of results and throughput.
Why Run-Time Knobs & Autotuning?

- In some applications internal knobs can be used at run-time to trade-off quality of results and throughput

- Autonomous Video-surveillance System

- Video Frame Rate

- Video Resolution

- Image 1: Camera setup in an autonomous surveillance system

- Image 2: Video frames being processed with different resolutions

- Image 3: Video frame rate gauge
Design-time Exploration

- Design-time exploration phase based on code profiling to define the relations between application parameters and performance metrics (such as energy, throughput and accuracy)

Operating Points combine application parameters and performance metrics

Working Modes represent max resource allocation associated to OPs
ARGO Application Autotuning Framework

Power consumption | Performance | Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

Application

OpenCL
ARGO Application Autotuning Framework

Goals

Rank

OP List

MAPE feedback loop

Power consumption
Performance
Accuracy

Application

OpenCL

Monitors

Autotuning Framework
ARGO Application Autotuning Framework

- Power consumption
- Performance
- Accuracy

1) Monitor

OpenCL

1) Monitor

OP List

Autotuning Framework

Goals +/-

Rank

Monitors

Application

11
ARGO Application Autotuning Framework

2) Analyze

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

Application

Power consumption  Performance  Accuracy
ARGO Application Autotuning Framework

Goals +/-

OP List

3) Plan

Monitors

Rank

Power consumption Performance Accuracy

3) Plan

Application

OpenCL
Application Autotuning Framework

Power consumption | Performance | Accuracy

Monitors → Goals +/- → Rank

OP List → Autotuning Framework

4) Execute

Application

OpenCL

Automation
Separation of Concerns

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

OpenCL

C++ source codes
Separation of Concerns

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

C++ source codes

OpenCL
Separation of Concerns

Adaption XML file

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

C++ source codes

Power consumption

Performance

Accuracy

OpenCL

C++ source codes

Knowledge XML file

Adaption XML file

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

C++ source codes

OpenCL
Separation of Concerns

- Power consumption
- Performance
- Accuracy

Adaption XML file

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

Adaptive Application

OpenCL

Adaptive Application Framework

Knowledge XML file

Monitors

Goals +/-

Rank

OP List

Adaption XML file
Orthogonality Concept: App Autotuning + RTRM

Target HW Platform

Platform OS

Requests

Resources

OpenCL

OpenCL

OpenCL
Orthogonality Concept: App Autotuning + RTRM
Orthogonality Concept: App Autotuning + RTRM

Run-Time Resource Manager

Platform OS

Target HW Platform

Resource Availability

http://bosp.dei.polimi.it
The Multi-View Case Study

2 eyes → third dimension
QoR: Disparity Error

Left camera

Right camera

reference disparity
QoR: Disparity Error

Left camera  Right camera

Reference disparity

Application Knobs

1 2 3

QoR Disparity Error
Experimental Setup

- **Target Platform**
  - Intel Xeon QuadCore CPU E5-1607 @ 3GHz & 8GB RAM
  - Linux 3.5 & OpenCL 1.2 run-time provided by Intel SDK 2013

- **Workload Definition**
  - Single application – single instance
    - Dynamically varying application requirements
  - Single application – multiple instances
    - Dynamic workload in terms of start time upon user request & Frame-rate goal demanded by user

- **Metrics of interest**
  - **Throughput**: Number of frames per second [Fps]
  - **Quality**: Normalized disparity error w.r.t reference
  - **Resources**: Percentage of CPU used by the application
Application Autotuning Tradeoffs

Graph 1: FPS (frames/sec) vs. Frame-rate goal (frames/sec)

Graph 2: ERR (%) vs. Frame-rate goal (frames/sec)
• First phase:
The application is processing the images by respecting the requirements.
Application Autotuning: Dynamic Adaptation (1)

- **First phase:**
  The application is processing the images by respecting the requirements.

- **Second phase:**
  There is a high priority task in the system requiring CPU resources.
Application Autotuning: Dynamic Adaptation (1)

- **First phase:**
The application is processing the images by respecting the requirements.

- **Second phase:**
There is a high priority task in the system requiring CPU resources.

- **Third phase:**
The task has finished releasing CPU resources.
• First phase:
The application is processing the images by respecting the requirements.
Application Autotuning: Dynamic Adaptation (2)

- Second phase: Threat detected
• Third phase:
The application is processing the images by respecting the requirements.
Multiple-Application: Run-Time Comparison Results

PowerDial
[ASPLOS2011]

ARGO

Throughput [fps]

D. Error [%]

CPU usage [%]

Time [s]

Throughput [fps]

D. Error [%]

CPU usage [%]

Time [s]
ARGO Overhead

Overhead [µs]

Number of Operating Points
Conclusions

- We proposed the ARGO autotuning framework to support runtime adaptivity in manycore architectures
  - To autonomously select and adapt at runtime the application configuration according to the requirements expressed as multi-objective constraints
  - To exploit performance/quality trade-offs at low overhead due to the combination of design-time and run-time techniques
- Most recent EU projects: FP7-2PARMA, FP7-HARPA (on-going) and FP7-CONTREX (on-going)