Design Space Exploration and Application Autotuning for Runtime Adaptivity in Multicore Architectures

Cristina Silvano
Politecnico di Milano
cristina.silvano@polimi.it
Outline

- Research challenges in multicore architectures
- Automatic design space exploration
- Application autotuning for runtime adaptivity
- Summary
RESEARCH CHALLENGES
Energy efficiency underlies all markets

- **Energy efficiency** is of paramount importance for all application markets (automotive, consumer, mobile, healthcare and beyond) and target systems spanning from sensors, cyber-physical systems, embedded systems up to servers and HPC systems.
Squeezing of computing cores

- 2005
  - 65 nm
  - 1.4 mm²
  - Source: ARM9 STMicroelectronics

- 2007
  - 45 nm

- 2009
  - 32 nm

- 2011
  - 22 nm

- 2013
  - 14 nm
... entering the multi/many-core era

Source: ARM9 STmicroelectronics
The design space in the age of multi/many-cores

- A wide range of architecture parameters must be tuned to find the best system configuration.
- Design space of the target architecture A should consider all possible configurations of each parameters $p_i$:
  
  $$A = S_{p1} \times S_{p2} \times \ldots \times S_{pn}$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Processors</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>#Threads</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>L1 I$ size</td>
<td>2K</td>
<td>16K</td>
</tr>
<tr>
<td>L1 D$ size</td>
<td>2K</td>
<td>16K</td>
</tr>
<tr>
<td>L1 I$ associativity</td>
<td>1w</td>
<td>8w</td>
</tr>
<tr>
<td>L1 D$ associativity</td>
<td>1w</td>
<td>8w</td>
</tr>
<tr>
<td>L2 $ size</td>
<td>32K</td>
<td>256K</td>
</tr>
<tr>
<td>L2 $ associativity</td>
<td>1w</td>
<td>8w</td>
</tr>
</tbody>
</table>

$\Rightarrow$ Large design space: $2^{18}$ (262,144) design points
FULL SEARCH
Can become quickly unfeasible

~10 minutes per simulation*

262,144 design points
\times
8 data sets
= 2,097,152 simulations

* Using a cycle-accurate simulator

Years

10
Research Challenges

1) What are the best tradeoffs to design many-cores in terms of multiple objectives such as energy and performance?

   - Need for automatic Design Space Exploration of many-core architectures
What are the barriers of further scaling?

- Transistor density increases ~2x every 2 years
- Frequency wall
- Power wall
- Utilisation wall

... the end of the Dennard scaling
... entering the dark silicon era
The Problem of Computation Offloading on Heterogeneous Parallel Platforms

Shift from the *power wall*... 
...to the *programmability wall*
Mobile Application Workloads

- Mobile users spend a high amount of time on a range of mobile applications*: 
  - 38% on web browsing and Facebook 
  - 32% on gaming 
  - 16% on audio, video and utility

* Source: Flurry Analytics
Mobile Application Workloads

- Short bursts of high intensity computation and power consumption
- Long periods of sustained high intensity
- Long-use low-intensity workloads

**Category 1**
Burst of High-intensity Workloads

**Category 2**
Sustained Performance at Thermal Limit

**Category 3**
Long-use Low-intensity Workloads

---

**Example:**
- Web Browsing
- Castlemaster
- Audio Playback

---

Measured on a Quad Cortex-A7 Symmetric Multiprocessing platform
Research Challenges

1) What are the best trade-offs to design many-cores in terms of multiple objectives such as energy and performance?
   - Need for automatic Design Space Exploration of many-core architectures

2) Are the system resources used efficiently?
   - Need for runtime resource management
Tunable Applications

- One or more application parameters, code transformations and code variants (*application knobs*) can be tuned at runtime
- Adaptivity to adjust the application behavior to the changing operating conditions, usage contexts and resource availability
Tunable Applications

- One or more application parameters, code transformations and code variants (application knobs) can be tuned at runtime

- Adaptivity to adjust the application behavior to the changing operating conditions, usage contexts and resource availability

- Approximate computing: output just needs to be “good enough” trading off accuracy and throughput
Research Challenges

1) What are the best trade-offs to design many-cores in terms of multiple objectives such as energy and performance?
   - Need for automatic Design Space Exploration of many-core architectures

2) Are we using the system resources efficiently?
   - Need for runtime resource management

3) How can we tune applications at runtime by using application knobs?
   - Need for application autotuning
AUTOMATIC DESIGN SPACE EXPLORATION
The multi-objective optimisation problem

- **Objective function:** To minimize both energy $\varepsilon(x)$ and execution time $\delta(x)$ of the target application on system configurations $x$:

$$\min_{x \in X} \omega(x), \quad \omega(x) = \begin{bmatrix} \varepsilon(x) \\ \delta(x) \end{bmatrix}$$

where $X$ is the design space.

- The solution is a set of tradeoff configurations $X_p \subseteq X$ known as Pareto set
Multi-objective exploration: Pareto Points

Multi-Objective Exploration: **best designs are not unique.**

Pareto points provide tradeoffs with respect to the multiple objectives.
The concepts behind the automatic DSE

Input Variables: architecture parameters that define the design space

Output Variables define the objective space

The black box generates the output values accordingly to the inputs.

The black box can be:

1) A simulator that models the system behavior and generates output values

2) A set of solvers that models the system behavior and estimates output values
MULTICUBE Explorer: What is it?

- MULTICUBE Explorer is an open source Multi-Objective Design Space Exploration (DSE) framework to tune multi-core architectures

- **Efficiency:** MULTICUBE Explorer is a design automation and acceleration tool to minimize the numbers of simulations to be executed during the DSE process

- **Flexibility:** MULTICUBE Explorer is a design optimization and exploration tool where you can easily plug-in your own simulator, optimization algorithms and machine learning techniques.

- MULTICUBE Explorer is **not** “yet another simulator” (there are already many simulators....)

www.multicube.eu
Multicube Explorer: Overview
Automatic Design Space Exploration: Efficiency

- **Efficiency** of automatic DSE process can be improved by:
  
  1. Minimizing the numbers of simulations to be executed by using *exploration heuristics* such as state-of-art evolutionary algorithms
  
  2. Speeding up simulations
  
  3. Simulating at higher abstraction levels
  
  4. Defining an *analytical response model* of the system behavior based on a subset of simulations to predict the unknown system response
1. Design of Experiments (DoEs):  
   To identify the experimentation plan: how to select the design points in the design space to be simulated

2. Optimisation Algorithms:  
   Meta-heuristics methods inspired by analogies with physics, or with biology to solve multi-objective optimization problems: simulated annealing, genetic algorithms, evolutionary strategies, etc.

3. Response Surface Modeling (RSM):  
   To use the set of simulated points to obtain an analytical model of the system behavior: linear regression, spline interpolation, artificial neural network, etc.
How to combine DoEs and Response Surface Models?

- **ReSPIR**: RSM-Support Iterative Pareto Refinement

```
DOE

Simulations

Stop?

Refining Simulations

Final Pareto Set

Intermediate Pareto Set

RSM

```
Benefits

- Why automatic design space exploration?
  - Faster exploration time
  - Better quality of results
Comparison Results: ReSPIR vs MOSA & NSGA-II

- Accuracy [%ADR]:
  - MOSA
  - NSGA-II
  - ReSPIR

- Design Space Analyzed [%]:
  - 217 design points (131,072)

- Accuracy in terms of Average Distance from Reference Set:
  - The lower the better approximated Pareto set

- MIPS-based shared memory multi-core
- Design space composed of $2^{17}$ design points (131,072)
Design-time optimization of OpenCL applications

The Multi-View Case Study:
The human eye stereo matching

2 eyes → third dimension
Quality of Results: Pixel Disparity Error

Left camera  Right camera

stereo-matching

Reference disparity map

Application Knobs

Disparity Error
Design Space Exploration (DSE)
Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application-specific and platform parameters

- Full search can become quickly unfeasible:
  - Multi-dimensional large design space
  - Slow simulation platforms


Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application-specific and platform parameters

- Full search can become quickly unfeasible:
  - Multi-dimensional large design space
  - Slow simulation platforms
Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application-specific and platform parameters
- Full search can become quickly unfeasible:
  - Multi-dimensional large design space
  - Slow simulation platforms
APPLICATION AUTOTUNING
Autonomous Video-surveillance System

- Application knobs can be used at runtime to trade-off the throughput and the quality of results

Video Frame Rate

Video Resolution
Multiple Cameras

Multiple instances of the same application

Autonomous Video-surveillance System

Video Frame Rate

Video Resolution
ARGO Application Autotuning

- **Key idea** is that most of the applications are dynamically configurable in terms of a set of tunable parameters, code transformations and code variants *(application knobs)* to trade-off accuracy and latency.
36
ARGO Application Autotuning

- **ARGO** is a light-weight application autotuning framework for multi/many-core platforms in an adaptive multi-application environment.
  - Combination of **design-time** and **run-time** techniques to create an effective way of "self-aware" computing with limited runtime overhead.
  - **Orthogonality** between application autotuning and runtime management of system resources.


Design-time exploration

- Design-time exploration phase based on code profiling to define the effects of application knobs and performance metrics (OPs) used to build a **prediction model** of the application behavior.

- Operating Points combine application knobs and performance metrics.

- Working Modes represent max resource allocation associated to OPs.
**MAPE feedback loop**

- **Application autotuning** enables self-optimization capabilities based on **Monitor-Analyze-Plan-Execute (MAPE) feedback loop***

---

ARGO Application Autotuning Framework

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

Application

OpenCL

Goals

Rank

OP List

Autotuning Framework
ARGO Application Autotuning Framework

Power consumption  Performance  Accuracy

Monitors
Goals +/-
Rank
OP List
MAPE feedback loop
OpenCL
Application

Autotuning Framework
ARGO Application Autotuning Framework

1) Monitor

Goals +/-

Rank

OP List

Autotuning Framework

Power consumption

Performance

Accuracy

OpenCL

Application
ARGO Application Autotuning Framework

Power consumption | Performance | Accuracy

2) Analyze

Monitors

Goals +/-

OP List

Autotuning Framework

Rank

Application

OpenCL

POLITECNICO DI MILANO
ARGO Application Autotuning Framework

Power consumption  Performance  Accuracy

Monitors

Goals +/-

OP List

Autotuning

3) Plan

Rank

Application

OpenCL

POLITECNICO DI MILANO
ARGO Application Autotuning Framework

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

OpenCL

4) Execute

Application
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

Diagram:
- Monitors
- Goals +/-
- Rank
- OP List
- Knowledge XML file
- OpenCL
- C++ source codes
Separation of Concerns

Goals

OP List

Adaption XML file

Rank

Knowledge XML file

C++ source codes

Power consumption Performance Accuracy

Monitors

Application

Autotuning Framework

Adaptation XML file

Knowledge XML file

C++ source codes
Separation of Concerns

- Power consumption
- Performance
- Accuracy

Adaption XML file

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

Adaptive Application

OpenCL

Adaptive Application

Knowledge XML file

Adaption XML file
Orthogonality Concept: App Autotuning & RTRM
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
Experimental Setup

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

- **Dynamic Workload Definition**
  - Single application reacting to external changes
  - Multiple instances of the same application

- **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

[ASAP2014]
Application Autotuning: Dynamic Adaptation (1)

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

![Graph showing CPU usage, throughput, and error percentages over time. The graph indicates a drop in CPU usage and an increase in throughput and error percentages at certain intervals. The text box on the right side of the graph highlights the second phase with the text: “Threat detected.”]
• Third phase:
The application is processing the images by respecting the requirements.
Multiple Application Dynamic Workload
ARGO + LINUX

PLAIN LINUX

35us overhead with 100 OPs

[ASAP2014]
ARGO+LINUX

35us overhead with 100 OPs

ARGO+RTRM

1ms overhead with 100 OPs

[ASAP2014]
Summary

- **MULTICUBE** Design Space Exploration framework and **ARGO** autotuning framework to support design-time multi-objective optimization and runtime adaptivity in manycore architectures.


- EU projects:
  - [www.multicube.eu](http://www.multicube.eu)
  - [www.2parma.eu](http://www.2parma.eu)
Next challenge: The Green500 scenario

- To reach the DARPA’s target of **20MW of Exascale supercomputers** projected to **2020**, current supercomputers must achieve an energy efficiency "quantum leap", pushing towards a goal of 50 GFlops/W.

The main goal of the ANTAREX project is to provide a breakthrough approach to express by a Domain Specific Language the application self-adaptivity and to runtime manage and autotune applications for green and heterogeneous High Performance Computing (HPC) systems up to the Exascale level. To be coupled with a radically new software stack capable of exploiting the benefits offered by heterogeneity to meet the scalability and energy efficiency required by the Exascale era.

**www.antarex-project.eu/**
Research Collaborations

POLITECNICO MILANO 1863

Research Collaborations with:

- ETH
- National Technical University of Athens
- Delft University of Technology
- RWTH Aachen University
- THALES
- Dompé
- gmv

Institute of Computing Technology, Chinese Academy of Sciences

IT4Innovations national supercomputing center
People