Application Autotuning for Energy Efficient Heterogeneous HPC Systems

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

- Research Challenges in the Exascale Era
- Application Autotuning
- ANTAREX Project
- Conclusions
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.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (PFlop/s)</th>
<th>Rpeak (PFlop/s)</th>
<th>Power (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center in Wuxi, China</td>
<td>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC</td>
<td>10,649,600</td>
<td>93.01</td>
<td>125.44</td>
<td>15.37</td>
</tr>
<tr>
<td>2</td>
<td>National Super Computer Center in Guangzhou, China</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P, NUDT</td>
<td>3,120,000</td>
<td>33.86</td>
<td>54.9</td>
<td>17.81</td>
</tr>
<tr>
<td>3</td>
<td>Swiss National Supercomputing Centre (CSCS), Switzerland</td>
<td>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc.</td>
<td>361,760</td>
<td>19.59</td>
<td>25.33</td>
<td>2.27</td>
</tr>
<tr>
<td>4</td>
<td>DOE/SC/Oak Ridge National Laboratory, US</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x, Cray Inc.</td>
<td>560,640</td>
<td>17.59</td>
<td>27.11</td>
<td>8.21</td>
</tr>
<tr>
<td>5</td>
<td>DOE/NNSA/LLNL, US</td>
<td>Sequoia - IBM BlueGene/Q, Power BQC 16C 1.60 GHz, Custom</td>
<td>1,572,864</td>
<td>17.17</td>
<td>20.13</td>
<td>7.89</td>
</tr>
</tbody>
</table>
Target Scenario

- Designing and tuning applications for energy-efficient High Performance Computing systems up to the Exascale era is an extremely challenging problem.

- Exascale supercomputers (reaching billions of billions FLOP per second) cannot be built by simply expanding the number of processing nodes and leveraging technology scaling, as power demand would increase unsustainably (up to hundreds of MW).

- To reach DARPA’s target of **20 MW of Exascale supercomputers** projected to the year 2020, current supercomputers (reaching up to 93 PetaFlop/s) must achieve an energy efficiency “quantum leap” to push toward **10x** energy efficiency from around **5 to 50 GFlops/W**
Target Scenario

- The **Green500 list** looks at **GigaFlops per Watt** as energy efficiency metric to rank supercomputers by their energy efficiency.

- According to the latest Green500 list (June 2017): the “most green” supercomputer TSUBAME 3.0 installed at the Tokyo Institute of Technology achieves **14 GigaFlops/W** during its 1.9PetaFlop/s per Linpack.

- Top 4 positions in Green500 are occupied by **heterogeneous systems** equipped with Intel Xeon processors and NVIDIA’s Tesla P100 GPUs.

- Next generation green HPC heterogeneous systems will integrate the latest NVIDIA Volta GV100 GPU to further accelerating the computation.
Target Scenario

- Dominance of heterogeneous systems in the Green500 list is expected to be a trend for the next coming years to reach the target of 20MW Exascale supercomputers.

- To reach the Exascale target, energy-efficient supercomputers need to be coupled with a radically new software stack capable of exploiting the benefits offered by heterogeneity at different levels:
  - At application level, to reduce the number of instructions per unit of computation;
  - At runtime level, to optimize the balancing and mapping of tasks and data on the heterogeneous resources;
  - At architecture level, to introduce energy-efficient and acceleration solutions exploiting heterogeneity and reconfigurability.
My Research Challenges

- Autotuning of HPC applications with respect to changing workloads, operating conditions and computing resources
- Providing programming models and domain specific languages to express self-adaptivity strategies and extra-functional requirements
- Monitoring the evolution of HPC platforms and exploiting heterogeneous computing by runtime resource and power 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/throughput/energy
Tunable Applications: Software Knobs

**Application Space**
- Application Parameters
  - Source to Source Code Transformations
  - Compiler Flags

**Target Independent Space**

**Target Dependent Space** (e.g. Intel Xeon & Xeon Phi)

**.bin**
AUTOMATIC DESIGN SPACE EXPLORATION
Objective function: To minimize both energy $\mathcal{E}(x)$ and execution time $\mathcal{D}(x)$ of the target application on system configurations $x$:

$$\min_{x \in X} \omega(x), \, \omega(x) = \begin{bmatrix} \mathcal{E}(x) \\ \mathcal{D}(x) \end{bmatrix}$$

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.
Full search simulation time

FULL SEARCH
Can become quickly unfeasible

~10 minutes per simulation*

simulation time on 4 parallel cores

10 Years

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

* Using a cycle-accurate simulator
1. **Design of Experiments (DoEs):**
   To identify the experimentation plan: how to select the design points in the design space to be simulated: random, full factorial, central composite design.

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.
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

1

2

3

Disparity Error
Design-time optimization of OpenCL applications
Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application parameters and platform parameters
- Full search can become quickly unfeasible due to huge multi-dimensional design space


Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application-specific and platform parameters
- Full search can become quickly unfeasible due to the huge multi-dimensional design space
Design-time optimization of OpenCL applications

- Parametric implementation:
  - Customization of application-specific and platform parameters
- Full search can become quickly unfeasible due to the huge multi-dimensional design space
APPLICATION AUTOTUNING
mARGOt 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.
mARGOt Application Autotuning

- mARGOt is a light-weight application autotuning framework for manycore 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


Runtime feedback loop

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

```
Monitor
```

```
Plan
```

```
Execute
```

```
Analyze
```

```
Modeling
application
knobs to metrics
```

```
Monitoring
some metrics
```

```
Configuration
selection
```

```
Tuning
application
knobs
```

```
IFTTT
```

**Heterogeneous Many-Core Platform**

mARGOt Application Autotuning

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework

Application

OpenCL

Goals

+/-

Rank

Monitors

OP List

Autotuning Framework

Application

mARGOt Application Autotuning

Power consumption  Performance  Accuracy

Monitors

Goals +/-

Rank

OP List

Autotuning Framework
mARGOt Application Autotuning

Goals

+/−

Rank

OP List

Monitors

Power consumption

Performance

Accuracy

Autotuning Framework

Application

OpenCL
Separation of Concerns

- Power consumption
- Performance
- Accuracy

Monitors → Goals +/- → Rank → OpenCL

- OP List
- Knowledge XML file
- C++ source codes
Separation of Concerns

- Power consumption
- Performance
- Accuracy

Adaption XML file

Monitors

Goals +/-

Rank

OP List

Knowledge XML file

C++ source codes

OpenCL

Autotuning Framework

Adaption 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
APPLICATION AUTOTUNING AND RUNTIME RESOURCE MANAGEMENT
Orthogonality Concept: App Autotuning & RTRM

Orthogonality Concept: App Autotuning & RTRM

- Target HW Platform
- Orthogonality Concept: App Autotuning & RTRM
- Platform OS
- Resource Availability
- Requests
- Resources
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

- **FPS [frames/sec]**
  - 1-core
  - 2-cores
  - 3-cores
  - 4-cores

- **ERR [%]**

- **Frame-rate goal [frames/sec]**

[ASAP2014]
**Application Autotuning: Dynamic Adaptation (1)**

- **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.
• **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.
Application Autotuning: Dynamic Adaptation (2)

- First phase:
  The application is processing the images by respecting the requirements.
• Second phase: Threat detected
• **Third phase:**
  The application is processing the images by respecting the requirements.
PLAIN LINUX

Multiple Application Dynamic Workload

[ASAP2014]
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]
Autotuning and Adaptivity Approach for Energy Efficient Exascale HPC Systems

http://www.antarex-project.eu/
Domain-Specific Language (DSL) to express:
- Runtime Adaptivity Strategies
- Complementary Knowledge and Execution Scenarios
- Compiler Optimization Strategies

- Autotuning
- Performance Improvements
- Compiler Optimizations

Accelerate:
- Productivity
- Performance
- Innovation

A biopharmaceutical application for accelerating drug discovery

Energy-Efficient Computing up to Exascale era:
- Scalable Monitoring
- Power Management

A self-adaptive navigation system for smart cities
Use Case 1: HPC Accelerated Drug Discovery System
Personalized Medicine will enable to *“treat the right patient with the right drug at the right dose at the right time.”* [FDA]

**HPC Accelerated Drug Discovery System**

- **Today**
  - Chemical Space: 6 Mio
  - Activity Prediction
  - Drug Likeness
  - Lead
  - Genotoxicity
  - Neurotoxicity
  - Cardiotoxicity
  - Adverse
  - Efficacy
  - Safe Drugs

- **Exascale**
  - Tangible Chemical Space: 300 Bio
  - Activity Prediction
  - Drug Likeness
  - Genotoxicity
  - Neurotoxicity
  - Cardiotoxicity
  - Adverse
  - Lead
  - Efficacy
  - Safer Drugs

**Need of HPC in Drug Discovery: HPC Molecular Simulations**
Developing energy and resource efficient algorithms
Using self-functionalities to adapt and scale-out the application

LiGen: Exascale-ready HPC Application

Chemical Space: 300 Bio
Peptide library: 120 Mio
Commercial libraries: 6 Mio
SafeInMan library: 9 K

Exascale HPC Virtual Screening
Molecular docking is a method to calculate the preferred position and shape of one molecule w.r.t a second one when bound to each other.

- **Geometric Docking**
  - Shape complementarity: Geometric matching search to find out compatible pairs and most suitable poses

- **Pharmacophoric Docking**
  - Molecular simulation: Exploration of a large energy landscape determined by chemical and physical interactions
Marconi: the most powerful supercomputer in Italy

- No. 14 in Top500 and No. 4 in Europe: Marconi Intel Xeon Phi:
  6.22 PetaFlops (Linpack performance) 10.83 PetaFlops (peak performance) with 241,808 cores. Site: Casalecchio di Reno, BO

- Marconi is the Cineca's Tier-0 system, co-designed by Cineca and Lenovo based on the Lenovo NeXtScale platform and Intel® Xeon Phi™ product family alongside with Intel® Xeon® processor E5-2600 v4 product family.

- In Nov. 2017, this system is planned to reach a computational power of about 20Pflop/s with future generation Intel Xeon processors (Sky Lakes).
Autotuning with mARGOt

- **time_to_solution**
- **#Cores/Nodes**

**PLANNER+ mARGOt**

- Ligand-DB MetaData
- **Low-Precision Step**
  - **High-Precision Step**
    - **#Iterations**
    - **Size Threshold**

**MINIAPP**

- Ligand-DB

Graph showing throughput per process (atoms/s) against score degradation [%].
Use Case 2: Self-adaptive Navigation System
Exploit synergies between client-side and server-side:

- Many drivers – many routing requests to HPC system
- Traffic status data sources
- Continuous update of traffic flow calculation
- Smart City Challenge
What is the time-dependent routing part of a navigation system?

- Module responsible for determining the **expected travel time**
- In the client-server navigation infrastructure, the **server side** evaluate accurate travel time with updated traffic information
- Based on **Monte Carlo Simulation (MCS)** over the **probabilistic speed profile** on each path segment
Autotuning with mARGOt (User & Data-aware)

Data Feature Extraction

Time + Path + Speed Profiles

Profiling Knowledge

OP List

Application Requirements

#Samples

MCS

Travel-Time Probability Distribution
http://www.antarex-project.eu/