LARA Tutorial

Plans for LARA in the context of the ANTAREX Project

Tiago Carvalho, Pedro Pinto, João Bispo, Ricardo Nobre, Luís Reis, and João M.P. Cardoso

University of Porto, FEUP, Porto, Portugal

April 20th, 2016

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

ANTAREX 10^{18}

SPeCS Special-Purpose Computing Systems, languages and tools

Outline

- Runtime adaptivity dimensions
- ANTAREX and LARA
- An example of ANTAREX + LARA
- Ongoing work
- Conclusions

Runtime Adaptivity Dimensions

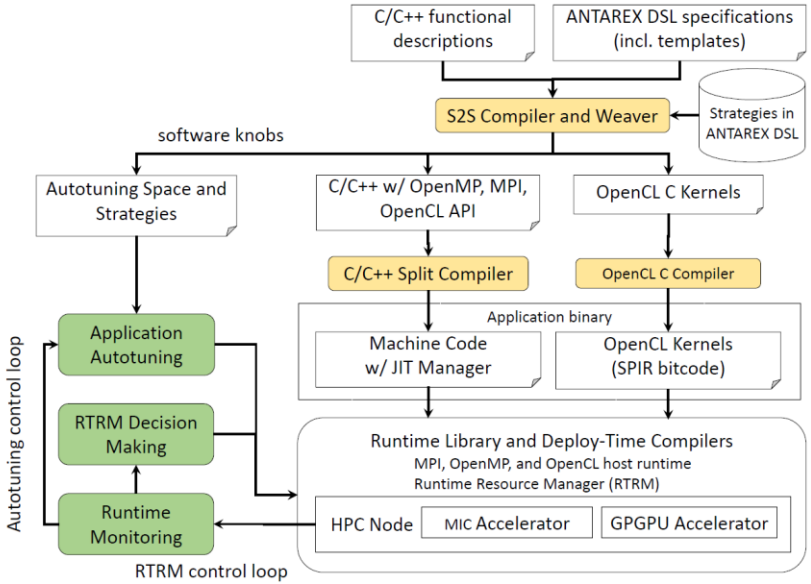
- From algorithm parameters to compiler and mapping optimizations
- Forms of adaptability include:
  - modifications to application parameters (attributes),
  - selection among different algorithms for solving the same problem,
  - different compiler optimizations for the same algorithm,
  - runtime strategies for partitioning and for mapping computations targeting hardware accelerators,
  - management of system resources

Extending LARA with native support for runtime adaptivity strategies

ANTAREX and LARA

AutoTuning and Adaptivity appRoach for Energy efficient eXascale HPC systems, FET-HPC, H2020 Project

- LARA as a vehicle to specify strategies for saving energy, increasing performance, etc.
- How?
  - Specifying strategies to optimize/specialize the code
  - Specifying strategies to refactor the code for autotuning and to control the autotuner



Example of a strategy supported by the ANTAREX Tool Flow:

- Create **multiple versions** of function "A"
- Insert calls to timers for **measuring the execution time** of the function
- **Substitute the call to the original function** with the possibility to execute one of the versions based on a parameter
- Instantiate an autotuner and **insert calls to the autotuner** and communication of execution time
- Use the **parameter output by the autotuner to select** between the versions of the function at runtime
- Apply to each version a **different optimization strategy**

Autotune every 1 sec

Silvano et al., ACM CF'2016

Ongoing Work

- Clang based source to source compiler controlled by LARA
- Code refactoring techniques to
  - increase performance, energy efficiency
  - support/help dynamic adaptivity schemes
  - expose autotuning opportunities (e.g., exposing parameters in some code transformations)
- Adding weaving support for offloading strategies (include code refactoring)
- Extending weaver with analysis for reporting information to be used by parallelization strategies (mainly OpenMP directives)
- Code refactoring and support for MPI
- Seamless integration of autotuning

Conclusions

- The LARA approach allows
  - To decouple functional and non-functional concerns
  - To express and capture transformation schemes and strategies
- Tool flows can be controlled and guided through aspects and strategies powered by LARA
- LARA leverages modularity, best practices and user expertise
- Experiments highlight developer's productivity and the usefulness of LARA
- Ongoing and Further Work will further enhance the usefulness of the LARA technology

Thank you! Questions?

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

ANTAREX

SPeCS
Special-Purpose Computing Systems, languages and tools