Clava + mARGOt = C/C++ to C/C++ Compiler and Runtime Autotuning Framework

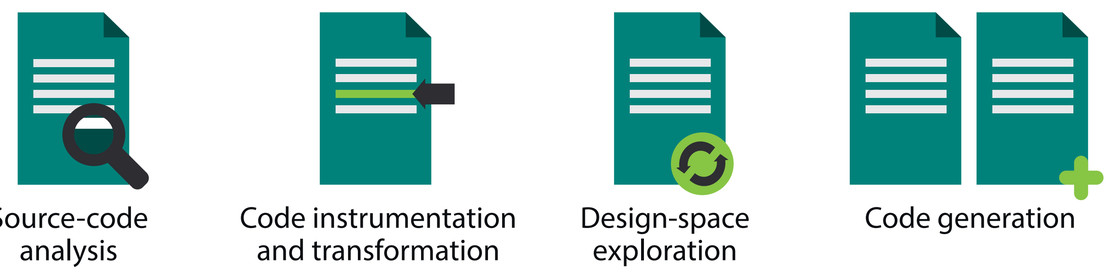
POLITECNICO MILANO 1863

U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

CLAVA

- Clang-based C/C++ source-to-source compiler
- Executes strategies written in LARA
- Highly modular and extensible
- Built-in APIs for integration and compilation

LARA in ANTAREX Tool Flow



Example usage scenarios

- AutoPar - Automatic parallelization using OpenMP
- HDF5 code generation - Automatic generation of HDF5 interface code
- OpenCL half-precision - Explore combinations of half-precision variables
- LAT - Explore parameterization of source code
- ANTAREX integration - Integration of toolflow components

Additional support features include

LaraDoc - Documentation generator

LaraUnit - Unit testing framework

Bundles - Pluggable implementations

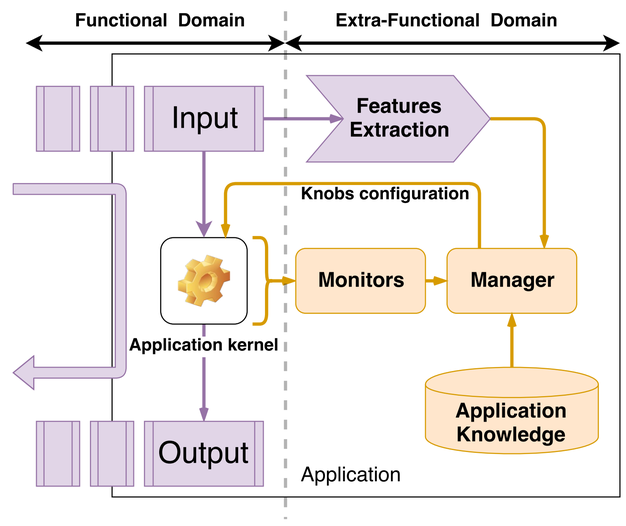
Online demo

specs.fe.up.pt/tools/clava

mARGOt

- Application-specific Run-Time manager, based in the MAPE loop (Monitoring, Analysis, Planning and Execution), with focus on Self-Optimization capabilities

- Enhances applications with an adaptation layer to continuously select the most suitable parameters to the application requirements defined as a constrained multi objective optimization problem



- The framework is implemented as a C++ library with an optional C interface

References

The autotuner framework and documentation:

https://gitlab.com/margot_project/core

Integration tutorial:

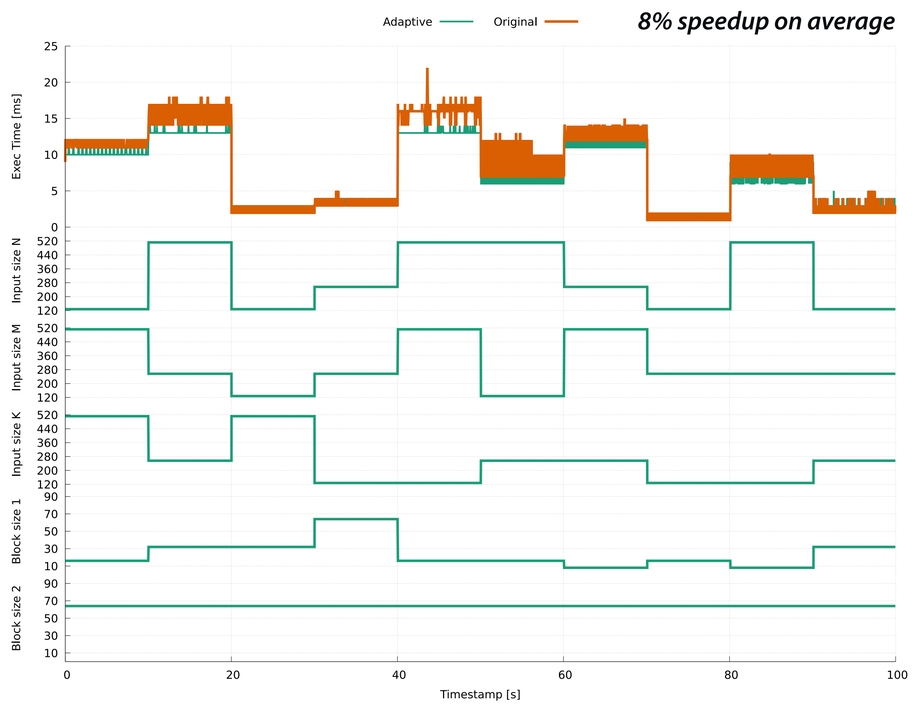
https://gitlab.com/margot_project/tutorial

Clava + mARGOt Demo

- Tiled matrix multiplication application

- Automated integration of the mARGOt autotuner
  - Clava with LARA strategies
  - Configuration file generation
  - Code instrumentation

- mARGOt runtime management
  - Monitoring of data features and metrics
  - Tile size control based on changing matrix sizes
  - Low-overhead adaptation layer



Tiled matrix multiplication application

Config.lara

```
var config = new MargotConfig(); // new mARGOt config
var matmul = config.newBlock('matmul'); // block
var timeMonitor = matmul.newTimeMonitor('timer');
matmul.addKnob('block_size_1', 'BS1', 'int');  // knob
…
```

config.xml

```
<!-- MONITOR SECTION -->
<monitor name="timer" type="TIME">
…
<!-- SW-KNOB SECTION -->
<knob name="block_size_1" var_name="BS1" var_type="int"/>
```

CodeGen.lara

```
select stmt.call{'matrix_mult_tiling'} end
apply
  codeGenInfo.update($call); // Update knobs before call
  codeGenInfo.monitorLog($call); // Monitor and log call
end
…
```

Dse.lara

```
select function{"main"}.body.call{"matrix_mult_tiling"} end
apply
  dseInfo.setScope($body); // DSE parameters scope
  dseInfo.setMeasure($call); // Collect metrics around call
end

// Add time metric
dseInfo.addTimeMetric('exec_time_us', TimeUnit.micro());
// Knob values
dseInfo.setKnobValues('block_size_1', 8, 16, 32);
…
```

matrix_mult.c

```
if(margot::matmul::update(BS1,BS2, N, M, K))
   margot::matmul::manager.configuration_applied();

margot::matmul::start_monitor();
matrix_mult_tiling(A, B, C, N, M, K, BS1, BS2);
margot::matmul::stop_monitor();
margot::matmul::log();
```

opList.xml

```
<parameters>
    <parameter name="block_size_1" value="8"/>
...
```

antarex-project.eu

ANTAREX is supported by the EU H2020 FET-HPC program under grant 671623

ANTAREX 10^{18}