Parallelware Analyzer: Static Code Analyzer for vectorization using OpenMP

Manuel Arenaz
manuel.arenaz@appentra.com
www.appentra.com
What is Appentra aiming to achieve?

- Boost the performance of C/C++ code by taking advantage of the parallelism chip manufacturers have put in multicore processors
- Introduce new developer tools to boost the performance of C/C++ code by exploiting parallel hardware
- Parallelware Analyzer, the first static code analyzer specialized in performance
- Leveraging the expertise of senior performance optimization software engineers to deliver faster applications for low-power multicore processors

Manuel Arenaz
CEO and Co-Founder

Boosting the performance of C/C++ code
Deep Performance Technology
A different Static Analysis Capability

● **Traditional Static Analysis**
  Focus on Bugs, Automating coding standard enforcement, or provide support for industry requirements like ISO 26262, IEC 61508, DO 178B – DOD 330

● **Issues**
  Lots of noise, difficult to configure, overpriced?

● **Nothing deals with C/C++ code performance**
  Code is often not written with Modern Hardware performance in mind
The first static code analyzer specialized in performance

Recommendations (PWR)
Boost performance and ensure best practices

Opportunities (OPP)
Sequential, vectorization, multi-threading and GPU offloading

Defects (PWD)
Find and fix bugs in parallel code and correctness verification

Remarks (RMK)
Proficient usage of tools

```
exampes/matmul pwreport src/main.c:15 --level 2 -- -I src/include
Compiler flags: -I src/include

FUNCTION BEGIN at src/main.c:matmul:6:1
  6: void matmul(size_t m, size_t n, size_t p, double **A, double **B, double **C) {
  LOOP BEGIN at src/main.c:matmul:15:5
    15:     for (size_t i = 0; i < m; i++) {
        [PWR]: src/main.c:15:5 'B' multi-dimensional array not accessed in row-major order
        [RMK]: src/main.c:18:28 avoid non-consecutive array access for variable 'A' to improve performance
        [RMK]: src/main.c:18:38 avoid non-consecutive array access for variable 'B' to improve performance
        [RMK]: src/main.c:18:25 avoid non-consecutive array access for variable 'C' to improve performance
      LOOP END
    FUNCTION END

FUNCTION BEGIN at src/main.c:main:24:1
  24: int main(int argc, char *argv[]) {
FUNCTION END
```
Scan source code without executing that code

List human-readable actionable recommendations on where and how to fix performance issues

Validate code against industry best practices for performance optimization

Integrate with Dev Tools and CI/CD frameworks

The first static code analyzer specialized in performance

**Recommendations (PWR)**
Boost performance and ensure best practices

**Opportunities (OPP)**
Sequential, vectorization, multi-threading and GPU offloading

**Defects (PWD)**
Find and fix bugs in parallel code and correctness verification

**Remarks (RMK)**
Proficient usage of tools
The first static code analyzer specialized in performance

Scan source code without executing that code

List human-readable actionable recommendations on where and how to fix performance issues

Validate code against industry best practices for performance optimization

Integrate with Dev Tools and CI/CD frameworks

Optimize performance on multicore CPUs (x86, Arm, Power)

Optimize performance on accelerator devices (GPU, FPGA)

Identify opportunities to enable performance techniques (sequential, vectorization, multi-threading, offload)

Refactor source code to actually implement parallelism

Recommendations (PWR)
Boost performance and ensure best practices

Opportunities (OPP)
Sequential, vectorization, multi-threading and GPU offloading

Defects (PWD)
Find and fix bugs in parallel code and correctness verification

Remarks (RMK)
Proficient usage of tools
Parallelware Analyzer 1.0 Vectorization Report

Parallelware Analyzer Demo for Canny Edge Detector

www.appentra.com
Thanks to the ORNL, NERSC, KAUST, PAWSEY and BSC for helping with the adoption of Parallelware Analyzer.

Thanks to the Horizon 2020 programme.

The EPEEC project has received funding from the European Union’s Horizon 2020 research and innovation programme under the grant agreement Nº 801051.

This project has received funding from the European Union's Horizon 2020 research and innovation program through grant agreement 801101.