Making OpenMP Easier with Parallelware Tools

Trainer & Analyzer

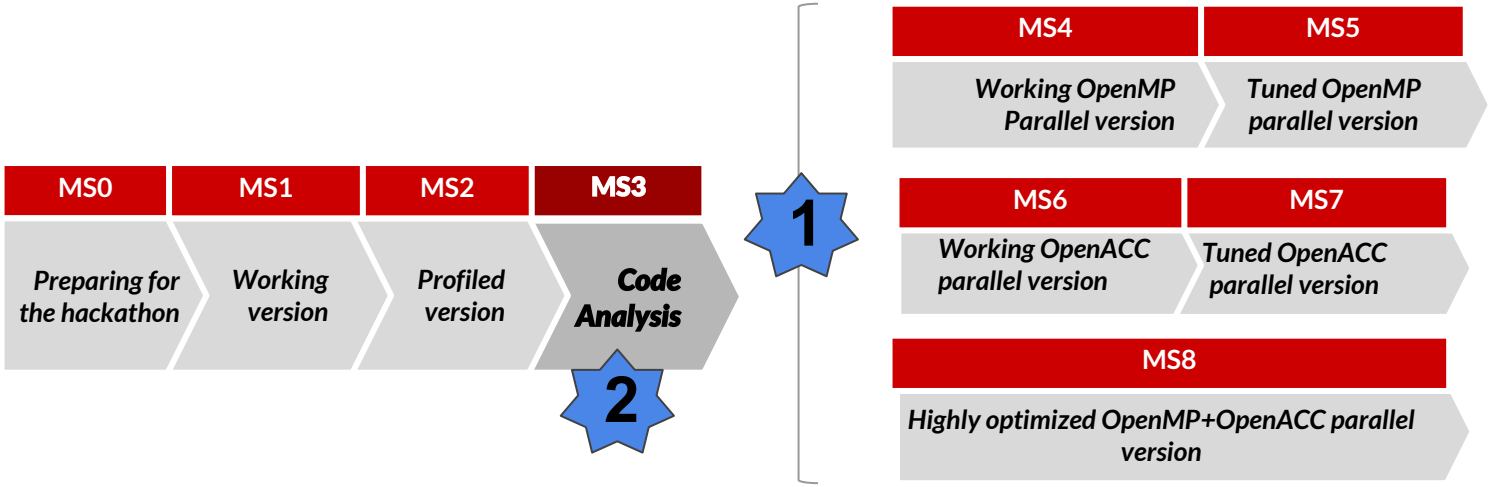
Manuel Arenaz
manuel.arenaz@appentra.com

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

appentra
make code parallel

OpenMP booth
14 November 2018 in Dallas

Biggest Parallelization Barriers?



1. Develop, test and benchmark several parallelization strategies in the current increasingly heterogeneous environment..
2. Automatic data scoping across procedure boundaries in the presence of complex in-memory data layouts

appentra
make code parallel

An interactive tool that acts as your mentor

> Tell me, I will forget,
> Show me, I may remember,
> Involve me, I will understand."

PARALLELWARE TRAINER

SC16 Salt Lake City, Utah hpc matters.

Emerging Technologies

appentra make code parallel

Parallelware Trainer (v1.0 Sep 2018)

Technical features:

- Identification of **parallelization opportunities**.
- Assistance with the introduction of correct **OpenMP** and OpenACC directives.
- **Correct data scoping**, including private/shared variables.
- Support for C programming language.
- Use any compiler and any build/compilation tool in **Windows, Linux and MacOS**.
- Develop, test and benchmark all within the same interface.

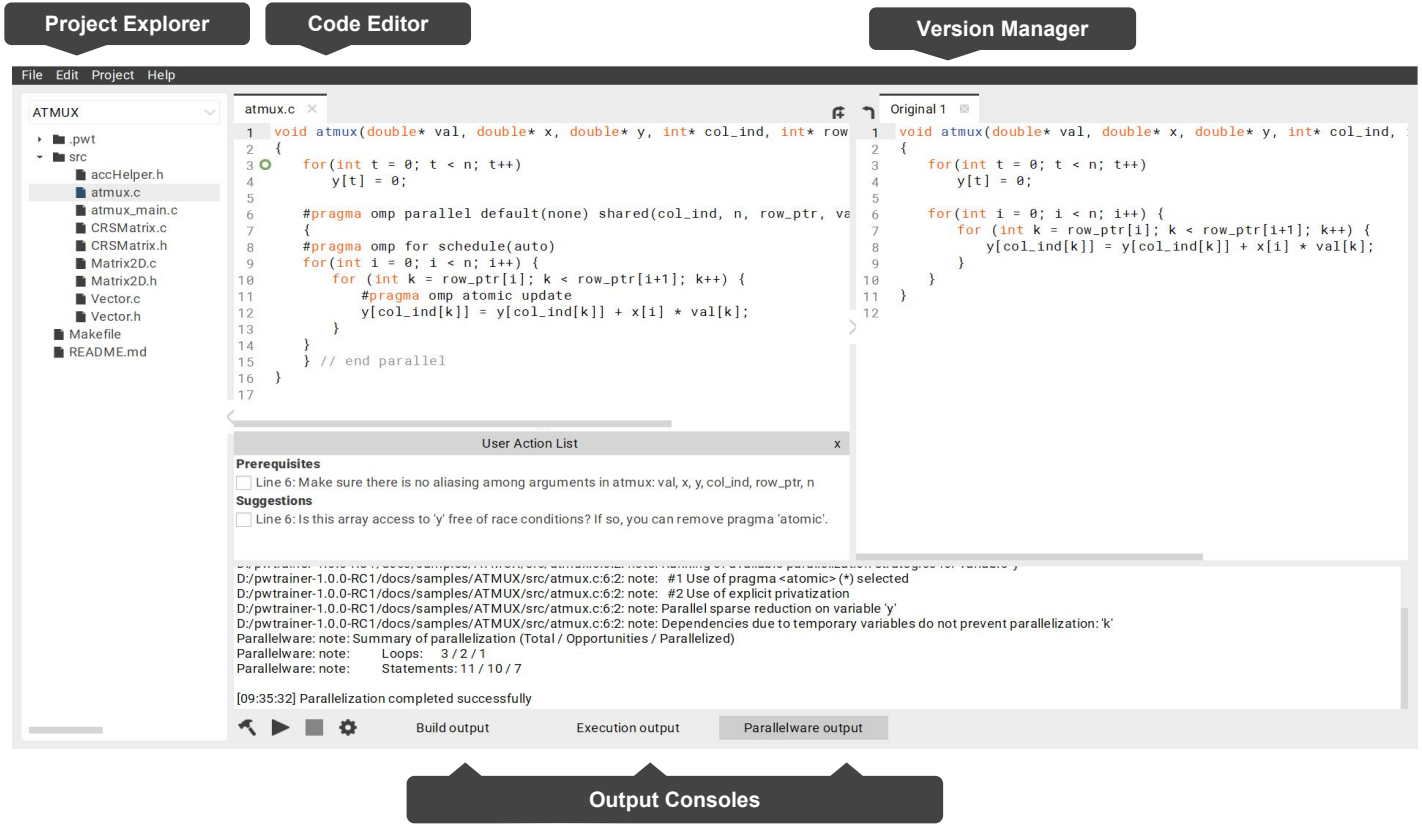
Benefits:

- **Faster, more effective learning**.
- **Work on realistic codes** rather than toy examples, including your own code.
- Reduced learning curve.
- Parallelize code within minutes.
- Immediate identification of where and how to parallelize.
- **Support for multithreading, offloading to GPUs**.

https://www.appentra.com/products/parallelware-trainer/

appentra
make code parallel

Parallelware Trainer (v1.0 Sep 2018)



PARALLELWARE TRAINER

appentra
make code parallel

A command-line reporting tool to improve productivity of HPC application developers

PARALLELWARE ANALYZER

SC18 Dallas, TX hpc inspires.

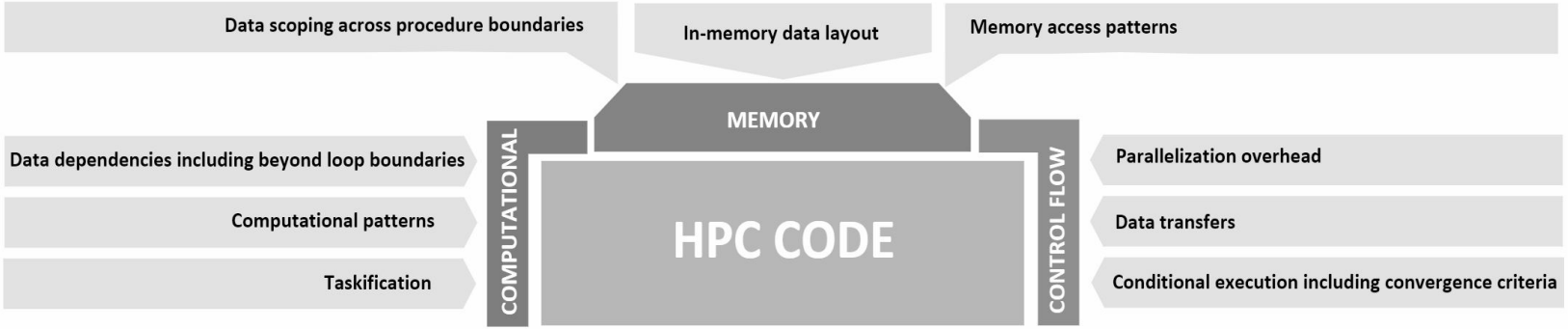
Emerging Technologies

appentra make code parallel

Parallelware Analyzer (Beta)

Command line tool leveraging Parallelware technology.

- Help to understand where and how to parallelize in real codes.
- Reports to facilitate **understanding the code from different perspectives**.
- Batch processing of files/directories of big code bases.



appentra
make code parallel

Parallelware Analyzer (Beta)

```
PWANALYZER(1)                          General Commands Manual                          PWANALYZER(1)

DESCRIPTION
       pwanalyzer provides different analyses to support code parallelization, reporting detailed information about the code, its data scoping, access
       patterns, parallelization methods, programming languages and data types.
SYNOPSIS
       pwanalyzer <analysis> [options] <source files/directories> [-- <compiler flags>]
ANALYSIS
       --overview    analyze programming languages (eg. C, C++, Fortran) and parallelization  methods (eg. OpenMP, MPI)
       --code        dumps the source code annotated with parallelization opportunities
       --functions   summarizes properties of the functions found in the code (eg. independent, number of parallel loops)
       --datalayout  provides information about all the built-in and provides information about all the built-in and
       --datascoping performs data scoping analysis
OPTIONS
       --noext                 skip external functions
       --brief                 do not print the verbose explanations about the meaning of all the fields in the tables
       --exclude <files/dirs>  skip the specified files and folders
       --simd | -multi         (functions, code) identify either innermost loops SIMD or outermost loops multithreading parallelization opportunities
       --name <name/regex>     (functions, datalayout) filter by name (accepts a regular expression)
       --openmp multi|offload  (datascoping) show OpenMP data scoping using the specified paradigm
       --openacc offload       (datascoping) show OpenACC data scoping using the specified paradigm
       --function <name/regex> (datascoping) filter by function name (accepts a regular expression)
       --loop <name/regex>     (datascopign) filter by loop name with syntax <file>:<function>:<line>:<column> (accepts a regular expression)
       -- <compiler flags>     gcc/clang-compliant flags required to build source code (must be last if present)
EXAMPLES
       $ pwanalyzer --overview src/
       $ pwanalyzer --functions --multi main.c
       $ pwanalyzer --code --multi main.c -- -I/usr/include/mpi -DCUSTOM_PARAM_ON
       $ pwanalyzer --datascoping --openacc offload --function 'compute.*' LULESHmk/
```

appentra
make code parallel

Parallelware Analyzer (Beta)

Automatic data scoping across procedure boundaries in codes using complex in-memory data layouts

```
▶ ./pwanalyzer --datascoping --openmp multi --openacc offload --function compute_energy_for_node ~/SC18/pwtrainer-samples/LULESHmk
6 total files found
  0 user-excluded files
  4 unknown type files
  2 source code files
   2 supported
    1 analyzable
    1 non-analyzable
   0 unsupported

[1/1] /home/jnovo/SC18/pwtrainer-samples/LULESHmk/src/lulesh_mk.c       SUCCESS

Loop                                    Variable Kind     Read/Write Temporary Pattern          OpenMP             OpenACC
--------------------------------------- -------- -------- ---------- --------- ---------------- ------------------ ------------------
lulesh_mk.c:compute_energy_for_node:32:5 r        scalar   rw
lulesh_mk.c:compute_energy_for_node:32:5 i        scalar   rw         x
lulesh_mk.c:compute_energy_for_node:32:5 TMP      array    wo                   sparse reduction shared/atomic(TMP) copyout/atomic(TMP)
lulesh_mk.c:compute_energy_for_node:32:5 pi       scalar   rw         x
lulesh_mk.c:compute_energy_for_node:32:5 tmp      scalar   rw         x
lulesh_mk.c:compute_energy_for_node:34:8 pi       scalar   ro                                      shared(pi)         copyin(pi)
lulesh_mk.c:compute_energy_for_node:34:8 i        scalar   rw
lulesh_mk.c:compute_energy_for_node:34:8 TMP      array    wo                   forall           shared(TMP)        copyout(TMP)
lulesh_mk.c:compute_energy_for_node:34:8 tmp      scalar   rw         x

Loop : loop name following the syntax <file>:<function>:<line>:<column>
Variable : name of the variable
Kind : variable datatype kind (scalar, pointer, array, dynarray, derived, other)
Read/Write : whether the variable is read-only ("ro"), write-only ("wo"), read-write ("rw") within the loop
Temporary : specifies whether the variable is internal to the loop (i.e. in C/C++ it is declared within the loop body)
Pattern : parallel pattern (read-only, forall, scalar reduction, sparse reduction)
OpenMP : OpenMP scoping (valid values: shared, private, reduction; special value: shared/atomic when atomic directive is also required)
OpenACC : OpenACC scoping (valid values: copy, copyin, copyout; special values: copy/atomic, copyout/atomic when atomic directive is also required)

1 files successfully analyzed and 0 failures in 124 ms
```

Making OpenMP Easier with Parallelware Tools

Trainer & Analyzer

Manuel Arenaz
manuel.arenaz@appentra.com

Unión Europea
Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"

appentra
make code parallel

OpenMP booth
14 November 2018 in Dallas