PARALLWARE TRAINER

LEARNING PARALLEL PROGRAMMING IS NOW FASTER AND EASIER!

WWW.PARALLWARE.COM

> **“Tell me, I will forget, Show me, I may remember, Involve me, I will understand.”**
>
> **Confucius**

Finding parallelism is more important than ever, but parallelizing a program still remains as an almost manual process. Thus, there is an urgent need for new tools that make parallel programming easier. Parallware Trainer is a new software tool for high-productivity HPC training using OpenMP and OpenACC.

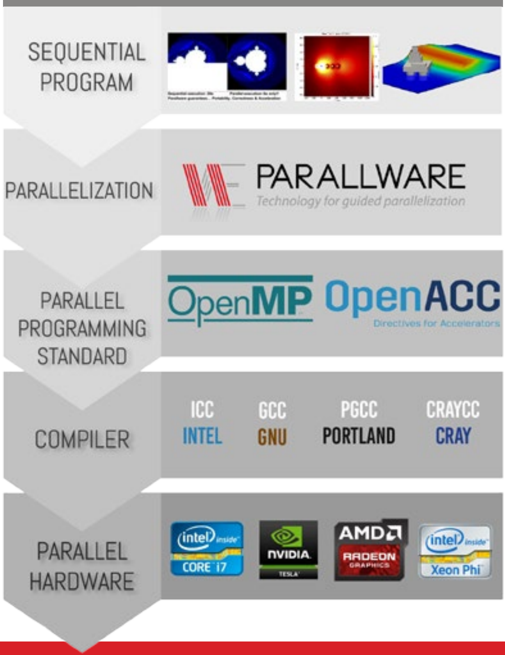
TECHNICAL FEATURES

- Interactive, real-time editor GUI
- Assisted code parallelization using OpenMP and OpenACC
- Transparent, local/remote execution and benchmarking
- Support for the C/C++ and Fortran programming languages
- Detailed report of the parallelism discovered in C code
- Support for multiple compilers.

BENEFITS

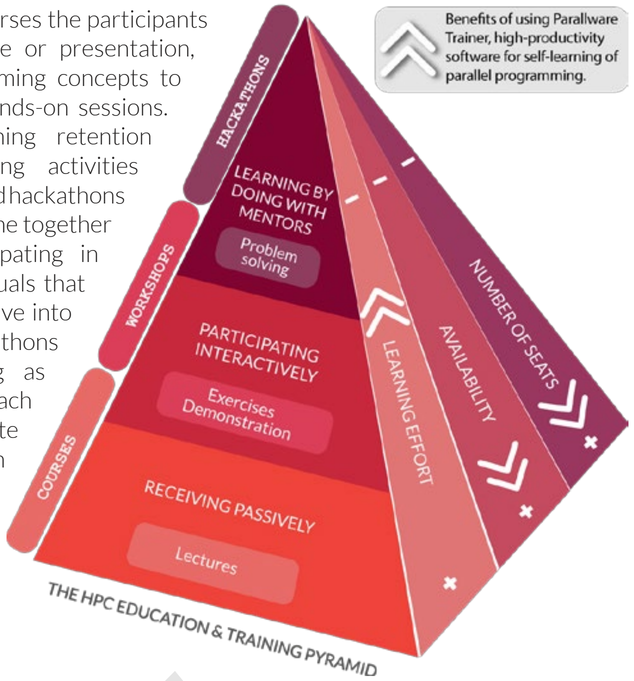
- Faster, more effective learning
- Reduced learning curve
- All-in-one learning tool for parallel programming
- Immediate use of parallel programming
- Support for multicore processors and GPUs

SC16 EMERGING TECHNOLOGIES
Salt Lake City, Utah hpc matters.



EXPERIENTIAL LEARNING

Education and training is organized today mostly around courses, workshops and hackathons. As shown in the pyramid, in courses the participants passively listen to a lecture or presentation, and apply parallel programming concepts to simple example codes in hands-on sessions. Workshops increase learning retention through interactive training activities between the participants. And hackathons are events where people come together to solve problems, participating in groups of about 2-5 individuals that take out their laptops and dive into their own problems. Hackathons allow experiential learning as participants work with a coach to learn through immediate practice in the optimization and parallelization of their own code. However, hackathons are expensive small events that train only a few people throughout the year.

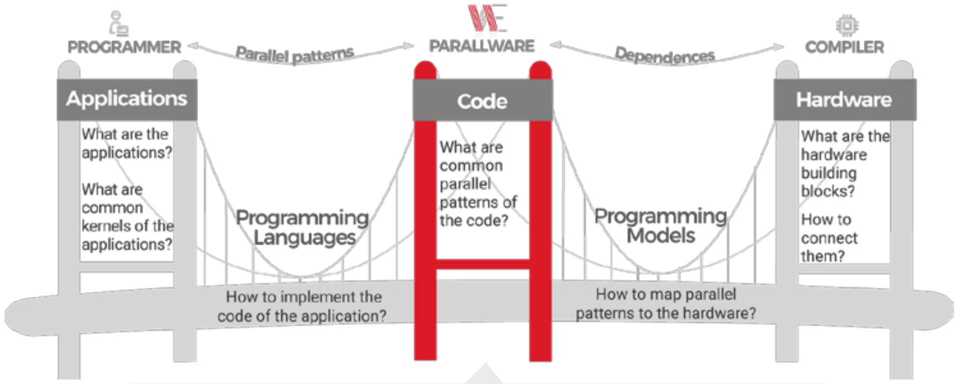

Benefits of using Parallware Trainer, high-productivity software for self-learning of parallel programming.

THE HPC EDUCATION & TRAINING PYRAMID

PARALLWARE TRAINER AIMS AT BRINGING THE BENEFITS OF WORKSHOPS AND HACKATHONS TO A BROADER STEM AUDIENCE

THE PARALLEL BRIDGE

Berkeley algorithmic **motifs**[1,2] describe the science of the applications. Motifs aim at helping in writing parallel programs so that it is as easy as writing codes for sequential computers. The bridge is used to illustrate that **Software is the problem #1** in bridging the gap between user Applications and the parallel Hardware industry. The new tower Code represents how Parallware approaches this software challenge. It highlights that the features of the code implemented by the programmer directly impact on the productivity of the parallelization process.


PROGRAMMER
Parallel patterns
PARALLWARE
Dependences
COMPILER
Applications
What are the applications?
What are common kernels of the applications?
Programming Languages
How to implement the code of the application?
Code
What are common parallel patterns of the code?
Programming Models
How to map parallel patterns to the hardware?
Hardware
What are the hardware building blocks?
How to connect them?

Best practices on parallel programming typically make coding recommendations, such as, to use stride-1 memory accesses and to prefer structures-of-arrays instead of arrays-of-structures.

PARALLEL PATTERNS USED IN PARALLWARE®

Production-level Compilers (e.g., Intel ICC, GNU GCC, NVIDIA PGI) are not of practical use to understand the failure reasons behind the parallelization of Applications. Compilers typically report the true/anti/output Dependences that prevent the parallel execution of a Code, being difficult for programmers to know how to rewrite their codes to enable parallelism. In contrast, Parallware reports the parallel patterns [3] found in the code, hiding the complexity of the dependences found by the compiler. The screenshot of Parallware Trainer [4] shows an example code from the motif *Sparse Linear Algebra*. The ATMUX code computes the product of a transposed sparse matrix and a vector, which corresponds to a *Parallel Sparse Reduction* in terms of Parallware parallel patterns.

parallel forall

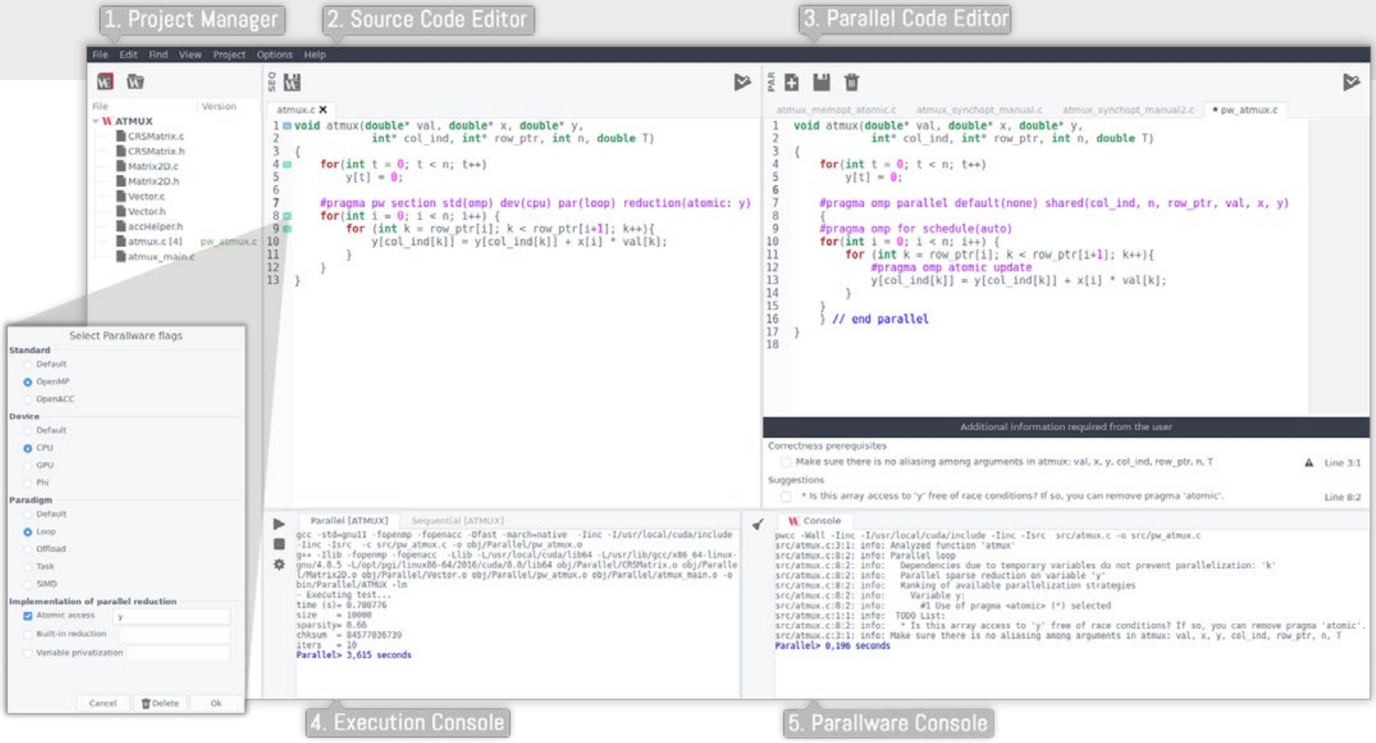
```
for ( j=0;  j<n;  j++)
{
     T= B [j];
     A[j] =T;
}
```

parallel scalar reduction

```
for ( j=0;  j<n;  j++)
{
     T = B [j];
     A + =T;
}
```

parallel sparse reduction

```
for ( j=0;  j<n;  j++)
{
     T = B [j];
     A [B [j] ] + =T;
}
```



Select Parallware flags

Standard
○ Default
● OpenMP
○ OpenACC

Device
○ Default
● CPU
○ GPU
○ Phi

Paradigm
○ Default
● Loop
○ Offload
○ Task
○ SIMD

Implementation of parallel reduction
☑ Atomic access: y
□ Built-in reduction
□ Variable privatization

```
void atmux(double* val, double* x, double* y,
           int* col_ind, int* row_ptr, int n, double T)
{
    for(int t = 0; t < n; t++)
        y[t] = 0;

    #pragma pw section std(omp) dev(cpu) par(loop) reduction(atomic: y)
    for(int i = 0; i < n; i++) {
        for (int k = row_ptr[i]; k < row_ptr[i+1]; k++){
            y[col_ind[k]] = y[col_ind[k]] + x[i] * val[k];
        }
    }
}
```

```
void atmux(double* val, double* x, double* y,
           int* col_ind, int* row_ptr, int n, double T)
{
    for(int t = 0; t < n; t++)
        y[t] = 0;

    #pragma omp parallel default(none) shared(col_ind, n, row_ptr, val, x, y)
    {
    #pragma omp for schedule(auto)
    for(int i = 0; i < n; i++) {
        for (int k = row_ptr[i]; k < row_ptr[i+1]; k++){
            #pragma omp atomic update
            y[col_ind[k]] = y[col_ind[k]] + x[i] * val[k];
        }
    }
    } // end parallel
}
```

Additional information required from the user

Correctness prerequisites

□ Make sure there is no aliasing among arguments in atmux: val, x, y, col_ind, row_ptr, n, T   ▲ Line 3:1

Suggestions

□ * Is this array access to 'y' free of race conditions? If so, you can remove pragma 'atomic'.   Line 8:2

```
gcc -std=gnu11 -fopenmp -fopenacc -Ofast -march=native  -Iinc -I/usr/local/cuda/include
-Iinc -Isrc  -c src/pw_atmux.c -o obj/Parallel/pw_atmux.o
g++ -Ilib -fopenmp -fopenacc  -Llib -L/usr/local/cuda/lib64 -L/usr/lib/gcc/x86_64-linux-
gnu/4.8.5 -L/opt/pgi/linux86-64/2016/cuda/8.0/lib64 obj/Parallel/CRSMatrix.o obj/Paralle
l/Matrix2D.o obj/Parallel/Vector.o obj/Parallel/pw_atmux.o obj/Parallel/atmux_main.o -o
bin/Parallel/ATMUX -lm
- Executing test...
time (s)= 0.700776
size    = 10000
sparsity= 0.66
chksum  = 84577036739
iters   = 10
Parallel> 3,615 seconds
```

```
pwcc -Wall -Iinc -I/usr/local/cuda/include -Iinc -Isrc  src/atmux.c -o src/pw_atmux.c
src/atmux.c:3:1: info: Analyzed function 'atmux'
src/atmux.c:8:2: info: Parallel loop
src/atmux.c:8:2: info:   Dependencies due to temporary variables do not prevent parallelization: 'k'
src/atmux.c:8:2: info:   Parallel sparse reduction on variable 'y'
src/atmux.c:8:2: info:   Ranking of available parallelization strategies
src/atmux.c:8:2: info:     Variable y:
src/atmux.c:8:2: info:       #1 Use of pragma <atomic> (*) selected
src/atmux.c:1:1: info:  TODO List:
src/atmux.c:8:2: info:   * Is this array access to 'y' free of race conditions? If so, you can remove pragma 'atomic'.
src/atmux.c:3:1: info: Make sure there is no aliasing among arguments in atmux: val, x, y, col_ind, row_ptr, n, T
Parallel> 0,196 seconds
```

References:

[1] K. Asanovic et al. 2009. A view of the parallel computing landscape. Commun. ACM 52, 10 (October 2009), 56-67. DOI: https://doi.org/10.1145/1562764.1562783

[2] T. Murphy. "Berkeley Motifs: Computation/Communication Algorithmic Patterns" (2013) PDF slides available at http://ncsilsu2013.wikispaces.com/file/view/BerkeleyMotifs.pdf

[3] Manuel Arenaz, Oscar Hernandez, Dirk Pleiter: The Technological Roadmap of Parallware and its Alignment with the OpenPOWER Ecosystem, International Workshop on OpenPOWER for HPC (IWOPH17) co-located with ISC17, 2017. https://doi.org/10.1007/978-3-319-67630-2

[4] Manuel Arenaz (Appentra), Sergio Ortega (UMA), Ernesto Guerrero (UMA), Fernanda Foertter (ORNL): "Parallware Trainer: Interactive Tool for Experiential Learning of Pararallel Programming using OpenMP and OpenACC" in Workshop on Education for High Performance Computing (EduHPC-17) co-located with SC17 (2017)

appentra
Software for guided parallelization

www.appentra.com
+34 881 015 556 | info@appentra.com

Appentra is a technology company established on 2012. It provides top quality software tools that allow extensive use of High Performance Computing (HPC) techniques in all application areas of engineering, science and industry. Appentra's target clients are companies and organizations that run frequently updated compute-intensive applications in markets like aerospace, automotive, civil engineering, biomedicine or chemistry.

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