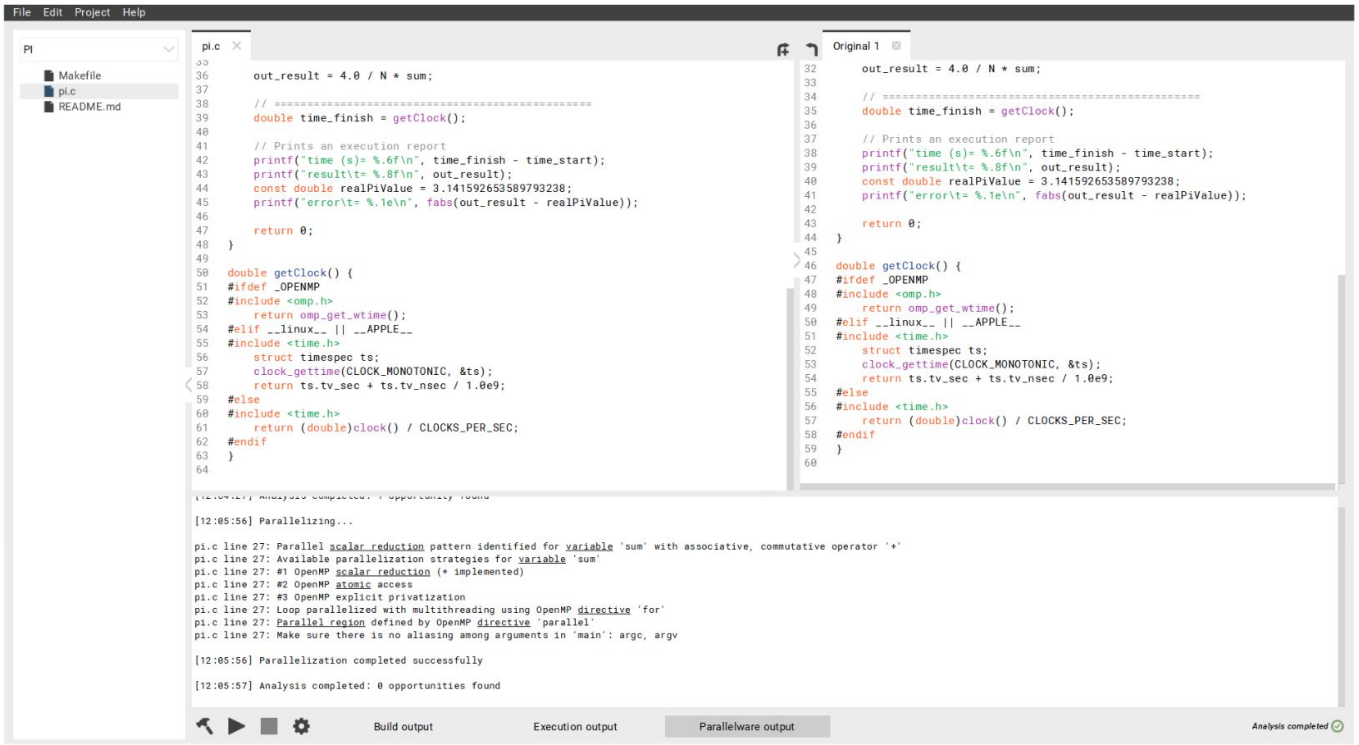
Parallelizing MATrix MULtiplication on the GPU with OpenMP/OpenACC

Goals:

- Build & run an OpenMP code on the GPU (for problem size N=5000)
- Build & run an OpenACC code on the GPU (for problem size N=5000)
  - Increase parallelism by collapsing loops
  - Choose the better memory layout for optimized memory transfers to the GPU

Parallelware Trainer



Run Parallelware Trainer on CORI

Step 1: Log in to CORI enabling X11 forwarding

```
$ ssh -X <your_login>@cori.nersc.gov
```

Step 2: Copy the sample codes to be used during the course

```
$ cp -a /global/common/cori_cle7/software/pwtrainer/Workshop-Oct19-examples.zip ~
```

Step 3: Prepare the environment by loading the appropriate modules

```
$ module purge && module load esslurm cuda gcc/8.1.1-openacc-gcc-8-branch-20190215 pgi/19.9 pwtrainer
```

Step 4 (*): Open an interactive session in a GPU node

```
$ salloc -t 60 -N 1 -c 8 -C gpu --gres=gpu:1 --mem=32GB --reservation=trainer -A nintern
```

(*) For CPU nodes, use --reservation=trainer_knl for the training
(*) Some people might need to use -A nstaff or -A m1759 instead of -A nintern

Step 5: Run the Parallelware Trainer tool from the GPU node

```
$ pwtrainer &
```

Memory layout versions

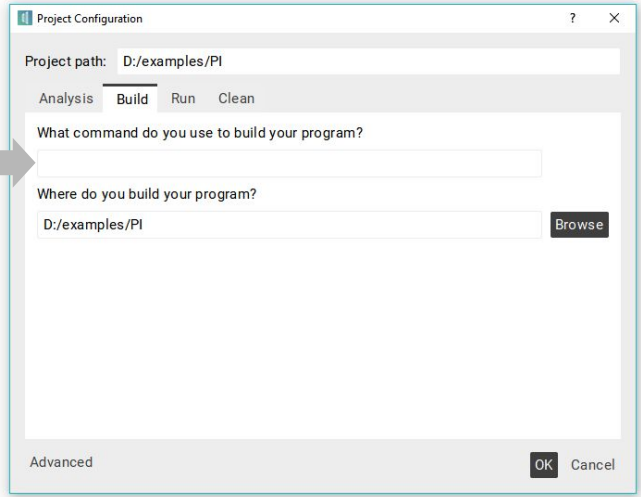
Three MATMUL versions differing in how 2D matrices are implemented using dynamic memory

Dynamic memory is preferred because it uses the **heap**,
unlike static memory which may consume the limited **stack** memory!

Version	Data structure	Memory layout	Array syntax
matmul_v1_dynarray2d_noncont.c	Dynamic array of pointers to dynamic 1D arrays (double**)	NON contiguous	A[i][j]
matmul_v2_dynarray2d.c	Dynamic array of pointers to locations of a dynamic 1D array containing the linearized data (double**)	Contiguous	A[i][j]
matmul_v3_dynarray1d.c	Dynamic 1D array containing the linearized data (double*)	Contiguous	A[i * cols + j]

Build in Parallelware Trainer

	OpenMP Enabling HPC since 1997	OpenACC More Science, Less Programming
GCC	`gcc -fopenmp -foffload=nvptx-none="-Ofast -lm -misa=sm_35" -o matmul matmul_v3_dynarray1d.c`	`gcc -fopenacc -foffload=nvptx-none="-Ofast -lm -misa=sm_35"-D_OPENMP -o matmul matmul_v3_dynarray1d.c`
PGI	`pgcc -mp -ta=tesla -Minfo=accel -o matmul matmul_v3_dynarray1d.c`	`pgcc -acc -ta=tesla -Minfo=accel -D_OPENMP -o matmul matmul_v3_dynarray1d.c`

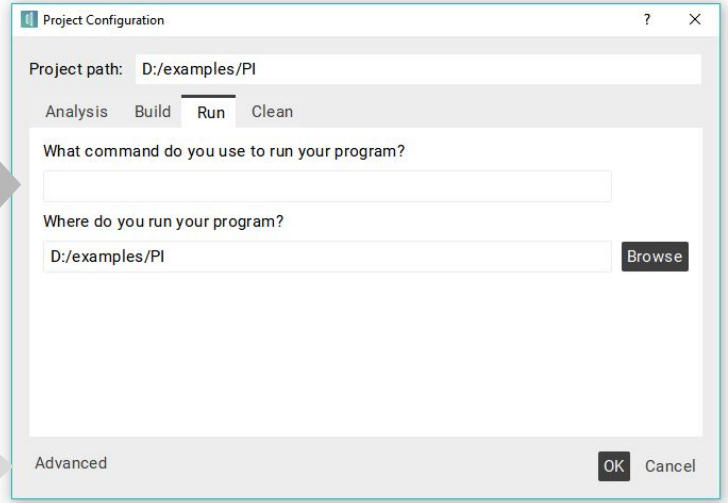


N.B. For Macs, check that your compiler supports OpenMP/OpenACC, and update with the appropriate compiler (e.g. gcc-mp-7)

Run in Parallelware Trainer

OpenMP Enabling HPC since 1997	`srun env OMP_NUM_THREADS=4 ./matmul 5000`
OpenACC More Science, Less Programming	`srun env ACC_DEVICE_TYPE=nvidia ./matmul 5000`

N.B. Instead of using the **env** command, you can add this variable by clicking the Advanced button.



Decomposition in parallel patterns

Code file "matmul_v3_dynarray1d.c"		**Pattern**			
Function	**Line**	**Forall**	**Scalar reduction**	**Sparse reduction**	**Convergence loop**
matmul()	22	C			
	32	C			
main()					