Parallelizing the CORAL LULESH microkernel

Goals:

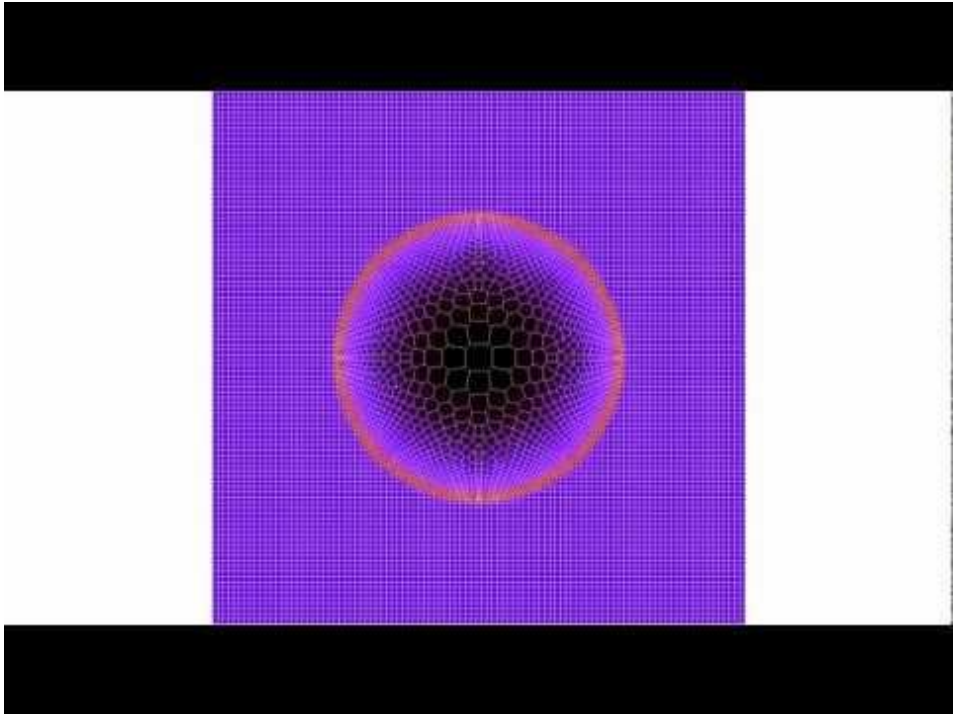
- Understand the code components
- Parallelize each component, including comparing performance of different implementations
- Combine parallel regions as required to improve performance
- Compare CPU and GPU versions
- Understand the data structure design used in the code and analyze pros/cons
- Minimize data transfers to the GPU

CORAL Benchmarks: LULESH

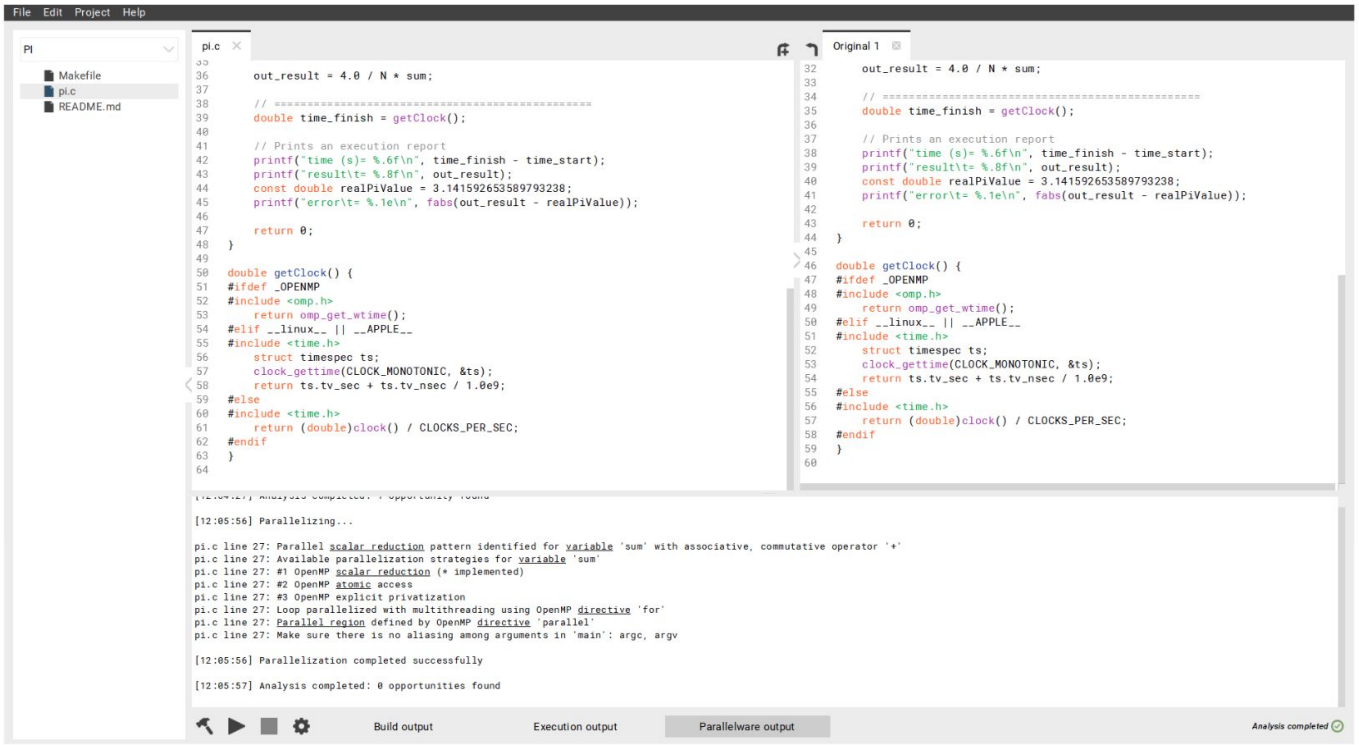
Livermore **U**nstructured **L**agrange
Explicit **S**hock **H**ydrodynamics

Part of a Physics Simulation software (ALE3D)

Models the propagation of a Sedov blast wave using Lagrangian hydrodynamics



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
```

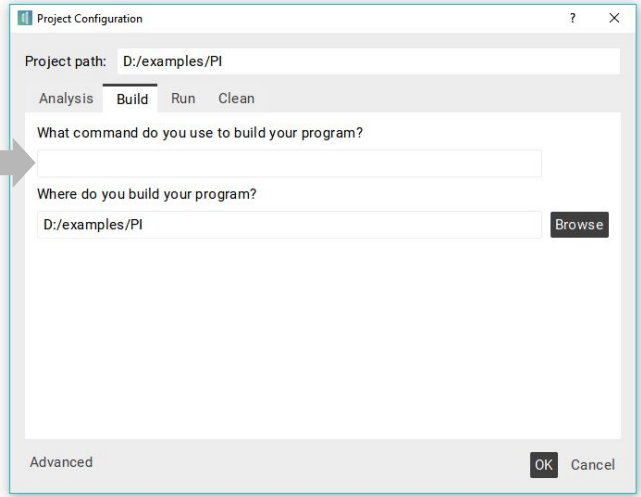
(*) For CPU nodes, use --reservation=trainer_knl for the training
(*) Some people might need to use -A nstaff or -A m1759 instead of -A nintern

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

```
$ pwtrainer &
```

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" -lm -o luleshmk luleshmk.c`	`gcc -fopenacc -foffload=nvptx-none="-Ofast -lm -misa=sm_35" -lm -D_OPENMP -o luleshmk luleshmk.c`
PGI	`pgcc -mp -ta=tesla -Minfo=accel -lm -o luleshmk luleshmk.c`	`pgcc -acc -ta=tesla -Minfo=accel -lm -D_OPENMP -o luleshmk luleshmk.c`

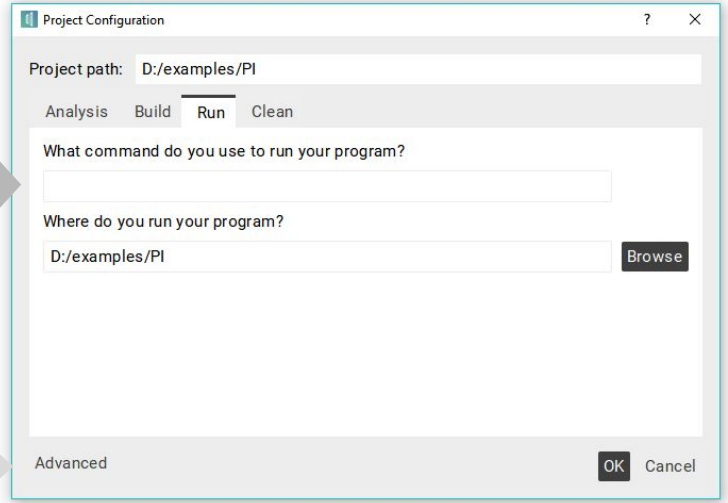


N.B. For Macs, check that your compiler supports OpenMP/OpenACC, and update with the appropriate compiler (e.g. gcc-mp-7)

Run in Parallelware Trainer

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

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



Profiling of LULESH microkernel

```
$ gcc -pg -o luleshmk luleshmk.c -lm
$ ./luleshmk
$ gprof ./luleshmk
```

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 56.23     24.62    24.62 223680000     0.00     0.00  CalcElemFBHourglassForce_workload
 22.76     34.59     9.97      932     0.01     0.01  ApplyMaterialPropertiesForElems_workload
 15.26     41.27     6.68 27960000     0.00     0.00  CalcElemVelocityGradient_workload
  2.26     42.26     0.99      932     0.00     0.03  CalcFBHourglassForceForElems
  2.24     43.24     0.98 27960000     0.00     0.00  CalcElemFBHourglassForce
  0.75     43.57     0.33 27960000     0.00     0.00  CalcElemVelocityGradient
  0.34     43.72     0.15        1     0.15    43.76  luleshmk
  0.09     43.76     0.04      932     0.00     0.01  CalcKinematicsForElems
  0.09     43.80     0.04                             frame_dummy
  0.00     43.80     0.00      932     0.00     0.01  ApplyMaterialPropertiesForElems
  0.00     43.80     0.00        2     0.00     0.00  calculate_checksum
  0.00     43.80     0.00        2     0.00     0.00  getClock
  0.00     43.80     0.00        1     0.00     0.00  Parameters_create
  0.00     43.80     0.00        1     0.00     0.00  Parameters_free
  0.00     43.80     0.00        1     0.00     0.00  VerifyAndWriteFinalOutput
```

How to verify correctness of LULESH

```
$ ./luleshmk
- Configuring the test...
- Executing the test...
gprof ./luleshmk
- Verifying the test...
Run completed:
   Problem size        =  30
   MPI tasks           =  1
   Iteration count     =  932
   Final Origin Energy = 1.000000e+00
   Number of nodes    = 27000
   Number of elements = 30000
   Number of regions  = 1
      Region 1 of size 30000
   Testing Plane 0 of Energy Array on rank 0:
        MaxAbsDiff   = 8.410000e+02
        TotalAbsDiff = 1.303550e+05
        MaxRelDiff   = 9.655568e-01

Elapsed time         =      71.00 (s)
Grind time (us/z/c)  =   2.821491 (per dom)  (  2.821491
overall)
FOM                  =  354.42254 (z/s)
```

Decomposition of LULESH into patterns

Code file "luleshmk.c"		**Pattern**				
Function	**Line**	**Forall**		**Scalar reduction**	**Sparse reduction**	**Convergence loop**
CalcElemFBHourglassForce_workload()	60			sum		
CalcElemFBHourglassForce()	73	hgfx				
CalcFBHourglassForceForElems()	**131**				**domain_m_fx**	
ApplyMaterialPropertiesForElems_workload()	187					
ApplyMaterialPropertiesForElems()	210 217					
CalcElemVelocityGradient_workload()	231					
CalcKinematicsForElems()	258					
luleshmk()	292					iter (loop index)
main()	349					
	358					
	365					
VerifyAndWriteFinalOutput()	485					