OpenMP Offloading Code on Summit

Wael R. Elwasif, Reuben Budiardja,
Swaroop Pophale, Thomas Papatheodore

Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory

Tutorial Code

- Repo : https://github.com/olcf/openmp-offload.git
- Simple Jacobi iterations with random initial conditions
  - C & Fortran versions of each variant
- Makefiles for the different compilers available
  - Invoked based on the loaded compiler module at compile time
- Example: Compile using GCC

```
[elwasif@login1.summit openmp-offload]$ module load gcc/11.1.0
[elwasif@login1.summit openmp-offload]$ cd C/0-serial/
[elwasif@login1.summit 0-serial]$ make
gcc -Ofast -fopenmp -Wl,-rpath=/sw/summit/gcc/11.1.0-2/lib64 -lm \
-foffload=nvptx-none="-Ofast -lm -latomic -misa=sm_35"  jacobi.c -o jacobi.C.gcc.exe
```

- Command line arguments : `num_cells max_iterations`
  - Except for code in **`5-openmp-gpu-implicit/`**

Jacobi iterations : Initialization

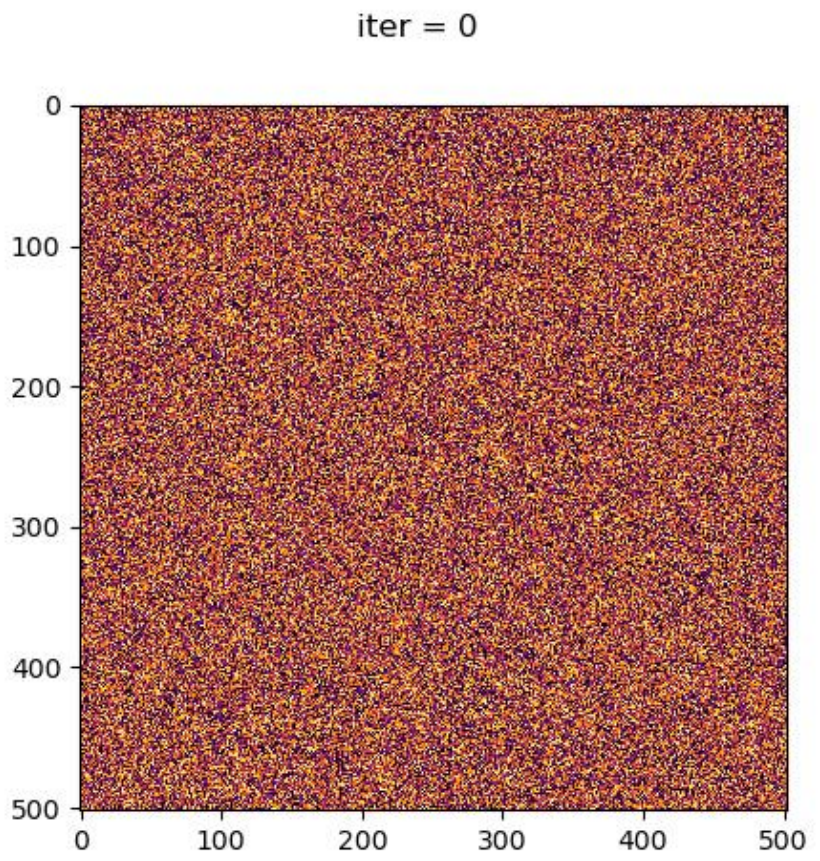
- Random seed generated and saved
- Regenerate the same problem for validation, or for runs using different configurations

```
void init(double *T) {

  static int first_time = 1;
  static int seed = 0;
  if (first_time == 1) {
    seed = time(0);
    first_time = 0;
  }
  srand(seed);

  for (unsigned i = 0; i <= n_cells + 1; i++) {
    for (unsigned j = 0; j <= n_cells + 1; j++) {
      T(i, j) = (double)rand() / (double)RAND_MAX;
    }
  }
}
```

Jacobi iterations : Serial version

```
// simulation iterations
  while (residual > MAX_RESIDUAL && iteration <= max_iterations) {
     // main computational kernel, average over neighbours in the grid
    for (unsigned i = 1; i <= n_cells; i++)
      for (unsigned j = 1; j <= n_cells; j++)
        T_new(i, j) =
            0.25 * (T(i + 1, j) + T(i - 1, j) + T(i, j + 1) + T(i, j - 1));

    // reset residual
    residual = 0.0;
    // compute the largest change and copy T_new to T
    for (unsigned int i = 1; i <= n_cells; i++) {
      for (unsigned int j = 1; j <= n_cells; j++) {
        residual = MAX(fabs(T_new(i, j) - T(i, j)), residual);
        T(i, j) = T_new(i, j);
      }
    }
    iteration++;
  }
  printf("Serial Residual = %.9lf\n", residual);
```

Jacobi iterations : 4-point filter



```
// simulation iterations
  while (residual > MAX_RESIDUAL && iteration <= max_iterations) {
     // main computational kernel, average over neighbours in the grid
    for (unsigned i = 1; i <= n_cells; i++)
      for (unsigned j = 1; j <= n_cells; j++)
        T_new(i, j) =
            0.25 * (T(i + 1, j) + T(i - 1, j) + T(i, j + 1) + T(i, j - 1));

    // reset residual
    residual = 0.0;
    // compute the largest change and copy T_new to T
    for (unsigned int i = 1; i <= n_cells; i++) {
      for (unsigned int j = 1; j <= n_cells; j++) {
        residual = MAX(fabs(T_new(i, j) - T(i, j)), residual);
        T(i, j) = T_new(i, j);
      }
    }
    iteration++;
  }
  printf("Serial Residual = %.9lf\n", residual);
```

The C/C++ Code variants

Directory	Description	Comments
`0-serial/`	Base serial version	
`1-openmp-cpu/`	OpenMP CPU only	
`2-openmp-gpu-teams/`	GPU: Teams only	Day 1
`3-openmp-gpu-parallel/`	GPU: Teams + Threads	Day 1
`4-openmp-gpu-data/`	GPU: Manage data movement	Day 2
`5-openmp-gpu-implicit/`	GPU: Implicit data movement	Day 2 – C++
`6-openmp-combined/`	All variants	

Similar Directory Structure for Fortran code

Summit OpenMP Offloading Compiler Support

- Vendor Provided & Supported:
  - XL
  - NVHPC Toolkit
- Community (Open Source):
  - LLVM
  - GCC

Summit OpenMP Offloading : Summary Table

Compiler C C++ Fortran	Module	Offloading Flags	Useful Flags	Useful Environment variables (verbose)
xlc C++ xlf	xl/16.1.1-10	-qsmp=omp -qoffload		
nvc nvc++ nvfortran	nvhpc/21.7	-mp=gpu -gpu=cc70	-Minfo=accel –Minfo=mp –Mino=loop	NVCOMPILER_ACC_NOTIFY
clang clang++ *flang*	llvm/12.0.0	-fopenmp \\ -fopenmptargets=nvptx64-nvidia-cuda \\ -Xopenmp-target \\ -march=sm_70		LIBOMPTARGET_INFO=-1
gcc g++ gfortran	gcc/11.1.0	-fopenmp	-foffload= “-lm -latomic”	GOMP_DEBUG=1

Note : The cuda module needs to be loaded for the LLVM clang compiler to target GPU offloading

Submitting Jobs On Sumit

- Use your own project ID
- Reservations from 2:00 – 4:30
  - **`#BSUB –U openmpWed`** on Wed
  - **`#BSUB –U openmpThu`** on Thu
- Sample batch script for 8 CPU threads
  - The **`–c`** and **`–bind packed:<x>`** argument needs to be (at least) the requested number of threads.

```
#!/bin/bash
# Begin LSF Directives
#BSUB -P PROJECT_ID
#BSUB -W 10:00
#BSUB -nnodes 1
#BSUB -U openmpWed
#BSUB -alloc_flags gpumps
#BSUB -J OMPtutorial
#BSUB -o OMPtutorial.%J
#BSUB -e OMPtutorial.%J

export OMP_NUM_THREADS=8
cd /PATH/TO/TUTORIAL/openmp-offload/C/1-openmp-cpu
date
jsrun -n1 –c $OMP_NUM_THREADS -g1 –bind packed:$OMP_NUM_THREADS
<EXECUTABLE>
```

See : https://docs.olcf.ornl.gov/systems/summit_user_guide.html#single-task-multiple-gpus-multiple-threads-per-rs

Experiments

- Compile and run the (GPU) code for the different compilers
  - Performance difference across compilers ??
  - Profile using nsight : https://docs.olcf.ornl.gov/systems/summit_user_guide.html#optimizing-and-profiling
- When is it profitable to offload to the GPU ?
  - Does it depend on the compiler ?
- Summit GPU's have 16 GB: What's the biggest problem you can solve?
  - Does the maximum problem size depend on the compiler?
- What's the impact of changing **`num_teams`** and **`thread_limit`** on performance
  - Can you figure out the default values used by the different compilers?