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 openmp-offload]$ make
```

• 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

```c
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

```c
// 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

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

*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

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

*Note: The cuda module needs to be loaded for the LLVM clang compiler to target GPU offloading*
Submitting Jobs On Summit

• 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](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?