Codee Training Series

April 25-26, 2023

Shift Left Performance

Automated Code inspection for Performance

April 24th, 2023

Walkthrough Exercise: Calculating π on the GPU with OpenMP/OpenACC

Goals

- Produce OpenACC version for GPU
- Produce OpenMP version for GPU
- Build & run an OpenMP code on the GPU (for problem size N=900000000)
- Build & run an OpenACC code on the GPU (for problem size N=900000000)

First consideration about using Codee at NERSC

- First, remember to load the Codee module
  ```
  $ module load codee
  ```
- The flag --help lists all the options available in the Codee command-line tools
  ```
  $ pwreport --help
  $ pwloops --help
  $ pwdirectives --help
  ```
- You can run Codee command-line tools on the login nodes (no need to run them on the compute nodes)
- Build and run the example codes on the compute nodes using the batch scripts
  - Scripts tuned to use the appropriate reservations: *codee_day1*, *codee_day2*
- Remember to check the open catalog of rules for performance optimization:

https://www.codee.com/knowledge/

The source code of PI

```
12    int main(int argc, char *argv[]) {
13        if (argc != 2) {
14                  printf("Usage: %s <steps>\n", argv[0]);
15                  printf("  <steps> controls the precision of the approximation.\n");
16                  return 0;
17        }
18
19        // Reads the test parameters from the command line
20        unsigned long N = atol(argv[1]);
21        printf("- Input parameters\n");
22        printf("steps\t= %lu\n", N);
23
24        printf("- Executing test...\n");
25        double time_start = getClock();
26        // ================================================
27
28        double out_result;
29
30        double sum = 0.0;
31        for (unsigned long i = 0; i < N; i++) {
32                  double x = (i + 0.5) / N;
33                  sum += sqrt(1 - x * x);
34        }
35
36        out_result = 4.0 / N * sum;
37
38        // ================================================
39        double time_finish = getClock();
40
41        // Prints an execution report
42        printf("time (s)= %.6f\n", time_finish - time_start);
43        printf("result\t= %.8f\n", out_result);
44        const double realPiValue = 3.141592653589793238;
45        printf("error\t= %.1e\n", fabs(out_result - realPiValue));
46
47        return 0;
48    }
```

Produce Codee Checks Report and follow suggestions!

```
$ pwreport --checks --verbose pi.c:main --include-tags gpu -- -lm -fast
Compiler flags: -lm -fast

[C] target compiler: <none> (Compiler Agnostic Mode)

CHECKS REPORT

pi.c:31:5 [PWR056]: consider applying offloading parallelism to scalar reduction loop
  Suggestion: use pwdirectives to automatically optimize the code
  Documentation: https://www.codee.com/knowledge/pwr056
  AutoFix:
        * Using OpenMP (recommended):
        pwdirectives --offload omp-teams --in-place pi.c:31:5 -- -lm -fast
        * Using OpenAcc:
        pwdirectives --offload acc --in-place pi.c:31:5 -- -lm -fast

SUGGESTIONS

  More details on the defects, recommendations and more in the Knowledge Base:
        https://www.codee.com/knowledge/

1 file successfully analyzed and 0 failures in 22 ms
```

Filter Codee output and focus on checks targeting GPU only

Focus on the suggested Codee commands to generate OpenMP and OpenACC source code

Generate code optimized for GPU with OpenMP directives

```
$ pwdirectives --offload omp-teams pi.c:31:5 -o pi_omp.c
[C] target compiler: <none> (Compiler Agnostic Mode)

Results for file 'pi.c':
  Successfully parallelized loop at 'pi.c:main:31:5' [using offloading]:
          [INFO] pi.c:31:5 Parallel scalar reduction pattern identified for variable 'sum' with associative, commutative operator '+'
          [INFO] pi.c:31:5 Available parallelization strategies for variable 'sum'
          [INFO] pi.c:31:5   #1 OpenMP scalar reduction (* implemented)
          [INFO] pi.c:31:5   #2 OpenMP atomic access
          [INFO] pi.c:31:5   #3 OpenMP explicit privatization
          [INFO] pi.c:31:5 Loop parallelized with teams using OpenMP directive 'target teams distribute parallel for'
Successfully created pi_omp.c

Minimum software stack requirements: OpenMP version 4.0 with offloading capabilities


$ cat -n pi_omp.c
…
          26        // ================================================
          27
          28        double out_result;
          29
          30        double sum = 0.0;
          31        #pragma omp target teams distribute parallel for shared(N) map(to: N) reduction(+: sum) map(tofrom: sum) schedule(static)
          32        for (unsigned long i = 0; i < N; i++) {
          33                  double x = (i + 0.5) / N;
          34                  sum += sqrt(1 - x * x);
          35        }
          36
          37        out_result = 4.0 / N * sum;
          38
          39        // ================================================

…
```

Just copy & paste the suggested invocation of pwdirectives, which will rewrite the code for you adding OpenMP directives

Note: source code edited "in-place" by default" and in this example we are using “-o” to write a separate source code file.

By default the OpenMP generated code:
- offloads the computation with "target teams"
- manages data transfers with “map”
- splits workload with “schedule(static)”

Generate code optimized for GPU with OpenACC directives

```
$ pwdirectives --offload acc pi.c:31:5 -o pi_acc.c
[C] target compiler: <none> (Compiler Agnostic Mode)

Results for file 'pi.c':
  Successfully parallelized loop at 'pi.c:main:31:5' [using offloading without teams]:
          [INFO] pi.c:31:5 Parallel scalar reduction pattern identified for variable 'sum' with associative, commutative
          [INFO] pi.c:31:5 Available parallelization strategies for variable 'sum'
          [INFO] pi.c:31:5   #1 OpenACC scalar reduction (* implemented)
          [INFO] pi.c:31:5   #2 OpenACC atomic access
          [INFO] pi.c:31:5 Parallel region defined by OpenACC directive 'parallel'
          [INFO] pi.c:31:5 Loop parallelized with OpenACC directive 'loop'
          [INFO] pi.c:31:5 Data region for host-device data transfers defined by OpenACC directive 'data'
Successfully created pi_acc.c

Minimum software stack requirements: OpenACC version 2.0 with offloading capabilities

$ cat -n pi_acc.c
...
    26        // ================================================
    27
    28        double out_result;
    29
    30        double sum = 0.0;
    31        #pragma acc data copyin(N) copy(sum)
    32        #pragma acc parallel
    33        #pragma acc loop reduction(+: sum)
    34        for (unsigned long i = 0; i < N; i++) {
    35            double x = (i + 0.5) / N;
    36            sum += sqrt(1 - x * x);
    37        }
    38
    39        out_result = 4.0 / N * sum;
    40
    41        // ================================================
...
```

In a similar manner, for OpenACC just copy & paste the suggested invocation of pwdirectives, which will rewrite the code for you adding OpenACC directives.

Note: source code edited "in-place" by default" and in this example we are using “-o” to write a separate source code file.

By default the OpenACC generated code:
- offloads the computation with "parallel"
- manages data transfers with “data copy”

Note: OpenACC provides a more elegant solution to manage data transfers for double** data types.

Benchmarking on Perlmutter @NERSC using Nvidia toolchain

Launch script “launch.sh”

```
#!/bin/bash
#SBATCH -A ntrain8
#SBATCH --reservation=codee_day1_gpu
#SBATCH -C gpu
#SBATCH -J Codee_Matmul_C
#SBATCH -q regular
#SBATCH -t 0:10:00
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 128
#SBATCH --gpus-per-task=1

export SLURM_CPU_BIND="cores"
srun PI.sh
```

PI execution script “PI.sh”

```
…
module load PrgEnv-nvidia
rm -f pi pi_omp pi_acc

nvc pi.c -lm -fast -o pi
./pi 900000000

nvc -fast -mp -target=gpu -Minfo=mp -lm pi_omp.c -o pi_omp
./pi_omp 900000000

nvc -fast -acc -target=gpu -Minfo=acc -lm pi_acc.c -o
pi_acc
./pi_acc 900000000
…
```

```
$ nvc pi.c -lm -fast -o pi
$ ./pi 900000000
- Input parameters
steps    = 900000000
- Executing test...
time (s)= 0.870196
result   = 3.14159265
error    = 8.0e-15


$ nvc -fast -mp -target=gpu -Minfo=mp -lm pi_omp.c -o pi_omp
$ ./pi_omp 900000000
- Input parameters
steps    = 900000000
- Executing test...
time (s)= 0.421352
result   = 3.14159265
error    = 7.4e-14


$ nvc -fast -acc -target=gpu -Minfo=acc -lm pi_acc.c -o pi_acc
$ ./pi_acc 900000000
- Input parameters
steps    = 900000000
- Executing test...
time (s)= 0.248663
result   = 3.14159265
error    = 1.3e-14
```

OpenMP offload yields speedup 2.1x

OpenACC offload yields speedup 3.5x

codee

www.codee.com

info@codee.com

Subscribe: codee.com/newsletter/

USA - Spain

codee_com

company/codee-com/