CS 280 ASSIGNMENT 2:

Accelerators

NAME: Xiaopeng Xu KAUST ID: 129052

# Task 1:

* Annotate the rtm\_kernel code (original, uploaded for the first assignment) with openMP directives to get a multithreaded code. Apply changes to the code if needed.
* Compile your openMP code using the Intel compiler on the MIC system for the Intel Xeon CPU (Sandy bridge)
* Run the code with 1, 2, 4, 8 and 16 threads on the CPU and report the corresponding speedups in a **graph**
* Compile the code for the coprocessor, run it using 60, 120, 180 and 240 threads natively on the coprocessor and report the corresponding speedups in a graph

CPU speedup report:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. of threads (CPU) | 1 | 2 | 4 | 8 | 16 |
| Time (s) | 373.6395 | 188.4551 | 94.8865 | 48.3044 | 36.7293 |
| Speedup | 1.00 | 1.98 | 3.94 | 7.74 | 10.17 |

Figure 1. CPU speedup

Coprocessor speedup report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. of threads (mic0) | 60 | 120 | 180 | 240 |
| Time (s) | 71.6174 | 58.2261 | 59.5833 | 56.6691 |
| Speedup | 5.22 | 6.42 | 6.27 | 6.59 |

Figure 2. Coprocessor speedup

The directives I added are shown on figure 3. I added pragma for the two for loops in the time loop. Also I make i, x, y, z, lap private in order to make the threads independent, i is firstprivate in order to copy the value from outside.

CPU running shows that multiple threads give me a better performance. However, coprocessor speedup is not proportional to the number of threads. One possible reason is that there are race conditions among multiple threads. The CPU is 16 core with 2.60GHz frequency for each core. (Detailed information about the CPU is shown on attachment file cpuinfo.)

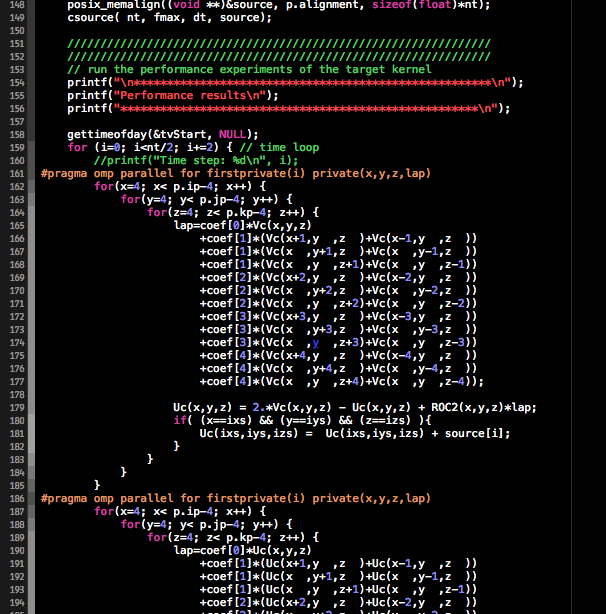


Figure 3. Directives added for OpenMP

# Task 2:

Annotate the rtm\_kernel code (original, uploaded for the first assignment) with openACC directives

* Compile it using the PGI compiler on the SMC system
* Run it using a fermi GPU and report the corresponding speedup
* Run it using a K20 GPU and report the corresponding speedup

GPU speedup report:

|  |  |  |
| --- | --- | --- |
| GPUs | Fermi | K20 |
| Time (s) | 24.3965 | 17.8206 |
| Speedup | 15.32 | 20.97 |

Figure 4. GPU speedup

The directives I added for OpenACC is shown on Figure 5. I copy the data outside of time loop, copy coeff, U3, source, and create/copy tempU1 and tempU2 (At first I just create tempU1, and tempU2. Then, in order to check the result I copied tempU1 and tempU2). I add two pragmas for each loop inside the time loop in order to achieve parallelism. Here collapse is used to make the 3 for loops collapse in serial codes, thus can easily be parallelized.

The running results shows that K20 gives me a better performance over Fermi.

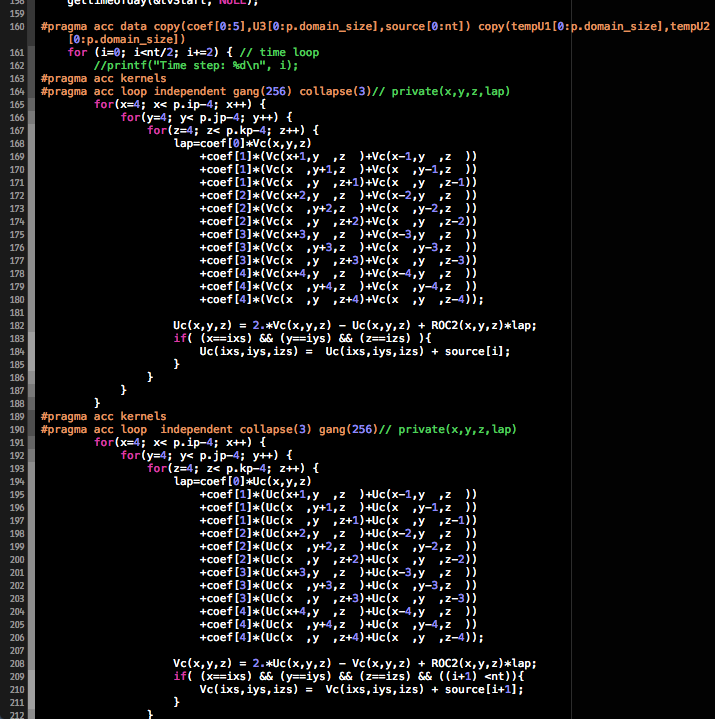


Figure 5. Directives added for OpenACC