CS639 HW1 Report

Computer Configuration

a) Computer used = MacBook Air 15-inch, M2, 2023
   Chip = Apple M2
   CPU = 8-core CPU with 4 performance cores and 4 efficiency cores
b) OS = 13.4 (22F2063)
   Compiler = clang++
c) compiled the code using a Makefile. The core instruction is:
   clang++ -fopenmp main.cpp Laplacian.cpp -o main
   Then I put this instruction into a Makefile:

```
# Define the compiler
CXX = clang++

# Compiler flags
CXXFLAGS = -fopenmp

# Source files
SOURCES = main.cpp Laplacian.cpp

# Object files
OBJECTS = $(SOURCES:.cpp=.o)

# Executable name
EXECUTABLE = main

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
	$(CXX) $(CXXFLAGS) $(OBJECTS) -o $@

.cpp.o:
	$(CXX) $(CXXFLAGS) -c $< -o $@

clean:
	rm -f $(OBJECTS) $(EXECUTABLE)
```

I installed two extra libraries for this code to be compiled: libomp and llvm via homebrew.

Performance Analysis

When using only 1 core, i.e. setting the OPM_NUM_THREADS = 1 in environment variable, the result is as follows:

```
alfredlong@Alfreds-MacBook-Air LaplacianStencil_0_10 % ./main
Running test iteration  1 [Elapsed time : 667.454ms]
Running test iteration  2 [Elapsed time : 626.444ms]
Running test iteration  3 [Elapsed time : 573.717ms]
Running test iteration  4 [Elapsed time : 574.474ms]
Running test iteration  5 [Elapsed time : 573.334ms]
Running test iteration  6 [Elapsed time : 575.286ms]
Running test iteration  7 [Elapsed time : 574.979ms]
Running test iteration  8 [Elapsed time : 574.128ms]
Running test iteration  9 [Elapsed time : 574.261ms]
Running test iteration 10 [Elapsed time : 575.357ms]
```

But when I switch to 8 cores, i.e. setting the said variable to 8, the result is this:

```
alfredlong@Alfreds-MacBook-Air LaplacianStencil_0_10 % ./main
Running test iteration  1 [Elapsed time : 172.868ms]
Running test iteration  2 [Elapsed time : 139.149ms]
Running test iteration  3 [Elapsed time : 154.308ms]
Running test iteration  4 [Elapsed time : 151.786ms]
Running test iteration  5 [Elapsed time : 152.24ms]
Running test iteration  6 [Elapsed time : 133.952ms]
Running test iteration  7 [Elapsed time : 141.14ms]
Running test iteration  8 [Elapsed time : 125.143ms]
Running test iteration  9 [Elapsed time : 127.577ms]
Running test iteration 10 [Elapsed time : 138.037ms]
```

This shows by using 8 cores it's giving an approximately 4x runtime boost.

So the effective bandwidth here is 1GB / 125ms = 8GB/s, where the peak bandwidth should be 100GB/s according to Apple. I did check I used OpenMP and set my environment variable to be theoretically maximum which is 8, but still I am getting less than 10% of the peak bandwidth. I think this might be something I can research in future.

Switching For-Loops Order

a) First in the Z direction then Y then X:

```
alfredlong@Alfreds-MacBook-Air LaplacianStencil_0_10 % ./main
Running test iteration  1 [Elapsed time : 1461.82ms]
Running test iteration  2 [Elapsed time : 1234.11ms]
Running test iteration  3 [Elapsed time : 1231.88ms]
Running test iteration  4 [Elapsed time : 1219.52ms]
Running test iteration  5 [Elapsed time : 1243.11ms]
Running test iteration  6 [Elapsed time : 1239.7ms]
Running test iteration  7 [Elapsed time : 1233.01ms]
Running test iteration  8 [Elapsed time : 1224.63ms]
Running test iteration  9 [Elapsed time : 1223.8ms]
Running test iteration 10 [Elapsed time : 1287.46ms]
```

We can see by switching the order upside-down, it is slowing down by 10x.

b) Switching to X -> Z -> Y:

```
alfredlong@Alfreds-MacBook-Air LaplacianStencil_0_10 % ./main
Running test iteration  1 [Elapsed time : 265.346ms]
Running test iteration  2 [Elapsed time : 236.179ms]
Running test iteration  3 [Elapsed time : 227.793ms]
Running test iteration  4 [Elapsed time : 240.433ms]
Running test iteration  5 [Elapsed time : 234.334ms]
Running test iteration  6 [Elapsed time : 306.411ms]
Running test iteration  7 [Elapsed time : 221.546ms]
Running test iteration  8 [Elapsed time : 230.787ms]
Running test iteration  9 [Elapsed time : 223.201ms]
Running test iteration 10 [Elapsed time : 228.226ms]
```

This is approximately 2x slower.

c) Comments:

I think it makes sense for the runtime to slow down dramatically. Because when switching the order, we are not accessing addresses that are closer to each other by the given order. Therefore, it might take more time for the OS to access memory then cache when running in the new order. Therefore the runtime is being slowed down.