CS639 Assignment 2 Report

Computer used = MacBook Air 15-inch, M2, 2023

Chip = Apple M2

CPU = 8-core CPU with 4 performance cores and 4 efficiency cores

How I compiled the code = clang++ -fopenmp main.cpp Laplacian.cpp ConjugateGradients.cpp PointwiseOps.cpp Reductions.cpp Utilities.cpp -O3 -o main.exe

Part1: Timing the kernels separately

Using all 8 cores’ results:

[Total Laplacian Time : 787.72ms]

[Total InnerProduct Time : 1944.83ms]

[Total Norm Time : 1211.07ms]

[Total Copy Time : 731.76ms]

[Total Saxpy Time : 5695.86ms]

[Main : 10376ms]

Using 1 core’s result:

[Total Laplacian Time : 1485.38ms]

[Total InnerProduct Time : 7591.54ms]

[Total Norm Time : 5034.51ms]

[Total Copy Time : 722.497ms]

[Total Saxpy Time : 5500.09ms]

[Main : 20338.5ms]

Part2: Merging the kernels

I merged Line2 into one kernel - Combined Kernel

float CombinedKernel(

float (&x)[XDIM][YDIM][ZDIM],

const float (&f)[XDIM][YDIM][ZDIM],

float (&p)[XDIM][YDIM][ZDIM],

float (&r)[XDIM][YDIM][ZDIM],

float (&z)[XDIM][YDIM][ZDIM])

{

float maxVal = 0.0; // For norm calculation

#pragma omp parallel for reduction(max:maxVal)

for (int i = 1; i < XDIM-1; i++) {

for (int j = 1; j < YDIM-1; j++) {

for (int k = 1; k < ZDIM-1; k++) {

// Compute Laplacian

float Lu = -6 \* x[i][j][k]

+ x[i+1][j][k] + x[i-1][j][k]

+ x[i][j+1][k] + x[i][j-1][k]

+ x[i][j][k+1] + x[i][j][k-1];

// Perform Saxpy (note that Lu is used instead of a separate z array)

r[i][j][k] = Lu \* (-1) + f[i][j][k]; // Assuming scale is -1 as in your example

// Update maxVal for Norm calculation

// Note: This is not the final value yet, just the max of this operation

maxVal = std::max(maxVal, std::abs(r[i][j][k]));

}

}

}

// maxVal now holds the maximum absolute value encountered during the operation,

// which corresponds to the Norm of the resultant array r.

return maxVal;

}

It’s called in this fashion:

// Algorithm : Line 2

// timerLaplacian.Restart(); ComputeLaplacian(x, z); timerLaplacian.Pause();

// timerSaxpy.Restart(); Saxpy(z, f, r, -1); timerSaxpy.Pause();

// timerNorm.Restart(); float nu = Norm(r); timerNorm.Pause();

timerCombinedKernel.Restart(); float nu = CombinedKernel(x, f, p, r, z); timerCombinedKernel.Pause();

And I also merged Line 8 into a new kernel - CombinedSaxpyNorm

float CombinedSaxpyNorm(

const float (&m) [XDIM][YDIM][ZDIM],

const float (&x)[XDIM][YDIM][ZDIM],

const float (&y)[XDIM][YDIM][ZDIM],

float (&z)[XDIM][YDIM][ZDIM],

const float scale) {

float result = 0.;

#pragma omp parallel for reduction(max:result)

for (int i = 1; i < XDIM-1; i++)

for (int j = 1; j < YDIM-1; j++)

for (int k = 1; k < ZDIM-1; k++) {

z[i][j][k] = x[i][j][k] \* scale + y[i][j][k];

result = std::max(result,std::abs(m[i][j][k]));}

return result;

}

It’s called like this:

// Algorithm : Line 8

// timerSaxpy.Restart(); Saxpy(z, r, r, -alpha); timerSaxpy.Pause();

// timerNorm.Restart(); nu=Norm(r); timerNorm.Pause();

timerCombinedSaxpyNorm.Restart(); nu = CombinedSaxpyNorm(r, z, r, r, -alpha); timerCombinedSaxpyNorm.Pause();

And the running time was reduced by around 1400ms with 8 cores.

[Total Laplacian Time : 832.769ms]

[Total InnerProduct Time : 2027.24ms]

[Total Norm Time : 0ms]

[Total Copy Time : 763.125ms]

[Total Saxpy Time : 3828.99ms]

[Total CombinedKernel Time : 16.2157ms]

[Total LaplacianAndInnerProduct Time : 0ms]

[Total CombinedSaxpyNorm Time : 1253.35ms]

[Main : 8726.75ms]

With only one core, the runtime was reduced by 1300ms as well.

[Total Laplacian Time : 1496.19ms]

[Total InnerProduct Time : 7852.69ms]

[Total Norm Time : 0ms]

[Total Copy Time : 777.977ms]

[Total Saxpy Time : 3663.15ms]

[Total CombinedKernel Time : 40.9209ms]

[Total LaplacianAndInnerProduct Time : 0ms]

[Total CombinedSaxpyNorm Time : 5191.12ms]

[Main : 19026.8ms]