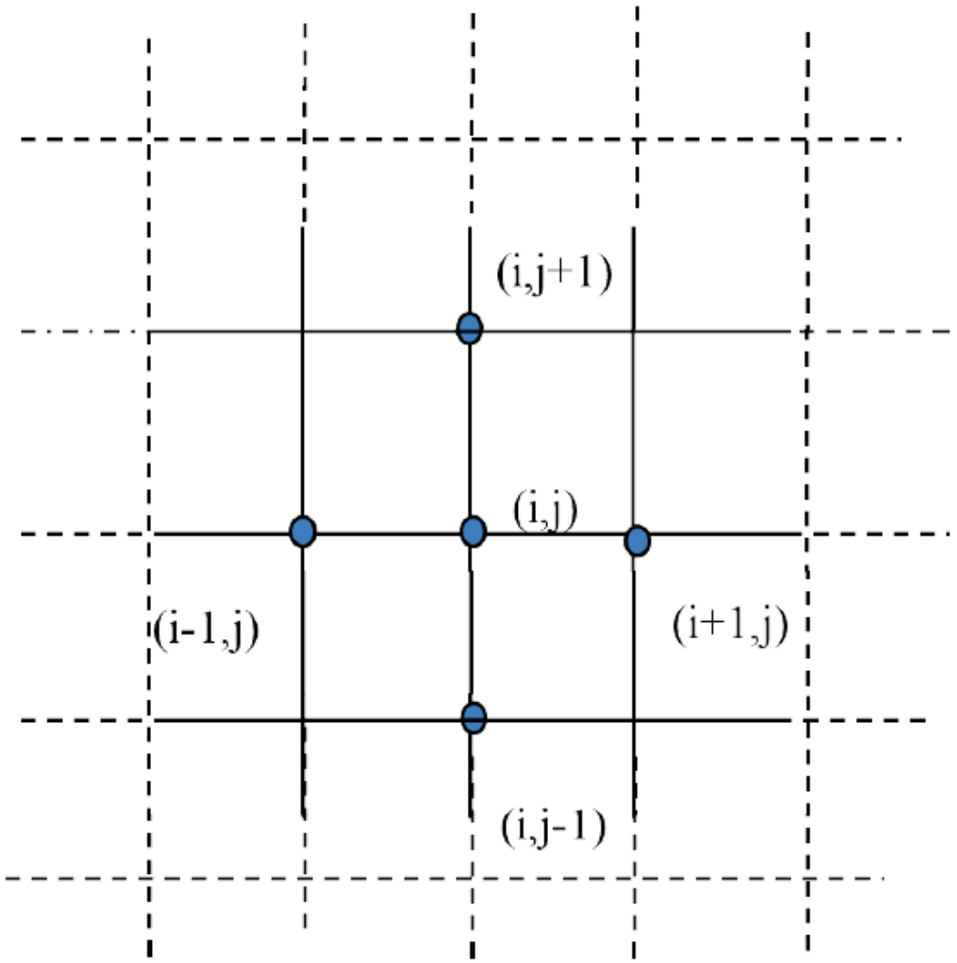
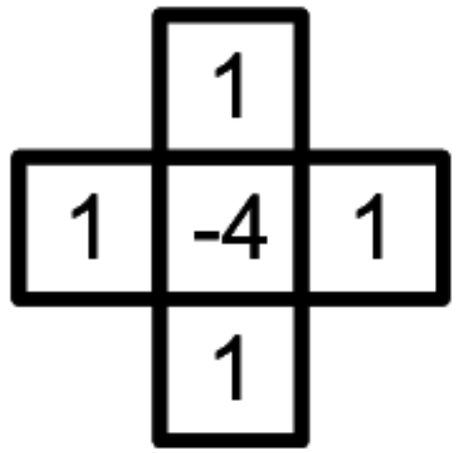
Case study #0

Laplacian Stencil Application
(Today : on 2D grid)





- All benchmarks discussed in class can be downloaded from the GitHub public repository

  https://github.com/sifakis/CS639S20_Demos.git

  (please report any access issues)

- Today's examples in Folder "LaplacianStencil"

- Subfolder of specific example listed on upper-right of each slide

- Execution times reported on a Single CPU Workstation with an Intel Xeon 6210U Processor
(20 cores @ 2.5Ghz)

- Peak Memory Bandwidth on this platform ~138GB/sec

- Peak Compute Bandwidth on this platform ~2.7TFLOPS

Timer Module (timer.h)

LaplacianStencil_XX_YY (all versions)

```
#pragma once

#include <chrono>
#include <cstring>
#include <iostream>

struct Timer
{
    using clock_t = std::chrono::high_resolution_clock;
    using time_point_t = std::chrono::time_point<clock_t>;

    time_point_t mStartTime;
    time_point_t mStopTime;

    void Start()
    {
        mStartTime = clock_t::now();
    }

    void Stop(const std::string& msg)
    {
        mStopTime = clock_t::now();
        std::chrono::duration<double, std::milli> elapsedTime = mStopTime - mStartTime;
        std::cout << "[" << msg << elapsedTime.count() << "ms]" << std::endl;
    }

};
```

Timer Module (timer.h)

LaplacianStencil_XX_YY (all versions)

```
#pragma once

#include <chrono>
#include <cstring>
#include <iostream>

struct Timer
{
    using clock_t = std::chrono::high_resolution_clock;
    using time_point_t = std::chrono::time_point<clock_t>;

    time_point_t mStartTime;
    time_point_t mStopTime;

    void Start()
    {
        mStartTime = clock_t::now();
    }

    void Stop(const std::string& msg)
    {
        mStopTime = clock_t::now();
        std::chrono::duration<double, std::milli> elapsedTime = mStopTime - mStartTime;
        std::cout << "[" << msg << elapsedTime.count() << "ms]" << std::endl;
    }

};
```

Benchmark launcher (main.cpp)

LaplacianStencil_0_[0-6]

```
#include "Timer.h"
#include "Laplacian.h"

#include <iomanip>

int main(int argc, char *argv[])
{
    using array_t = float (&) [XDIM][YDIM];

    float *uRaw = new float [XDIM*YDIM];
    float *LuRaw = new float [XDIM*YDIM];
    array_t u = reinterpret_cast<array_t>(*uRaw);
    array_t Lu = reinterpret_cast<array_t>(*LuRaw);

    Timer timer;

    for(int test = 1; test <= 10; test++)
    {
        std::cout << "Running test iteration " << std::setw(2) << test << " ";
        timer.Start();
        ComputeLaplacian(u, Lu);
        timer.Stop("Elapsed time : ");
    }

    return 0;
}
```

Benchmark launcher (main.cpp)

LaplacianStencil_0_[0-6]

```
#include "Timer.h"
#include "Laplacian.h"

#include <iomanip>

int main(int argc, char *argv[])
{
    using array_t = float (&) [XDIM][YDIM];

    float *uRaw = new float [XDIM*YDIM];
    float *LuRaw = new float [XDIM*YDIM];
    array_t u = reinterpret_cast<array_t>(*uRaw);
    array_t Lu = reinterpret_cast<array_t>(*LuRaw);

    Timer timer;

    for(int test = 1; test <= 10; test++)
    {
        std::cout << "Running test iteration " << std::setw(2) << test << " ";
        timer.Start();
        ComputeLaplacian(u, Lu);
        timer.Stop("Elapsed time : ");
    }

    return 0;
}
```

Allocate u & Lu so that they can be used as 2-dimensional arrays (i.e. u[56][67])

Benchmark launcher (main.cpp)

LaplacianStencil_0_[0-6]

```
#include "Timer.h"
#include "Laplacian.h"

#include <iomanip>

int main(int argc, char *argv[])
{
    using array_t = float (&) [XDIM][YDIM];

    float *uRaw = new float [XDIM*YDIM];
    float *LuRaw = new float [XDIM*YDIM];
    array_t u = reinterpret_cast<array_t>(*uRaw);
    array_t Lu = reinterpret_cast<array_t>(*LuRaw);

    Timer timer;

    for(int test = 1; test <= 10; test++)
    {
        std::cout << "Running test iteration " << std::setw(2) << test << " ";
        timer.Start();
        ComputeLaplacian(u, Lu);
        timer.Stop("Elapsed time : ");
    }

    return 0;
}
```

Use “Timer” class to time execution of your test(s)

Benchmark launcher (main.cpp)

LaplacianStencil_0_[0-6]

```
#include "Timer.h"
#include "Laplacian.h"

#include <iomanip>

int main(int argc, char *argv[])
{
    using array_t = float (&) [XDIM][YDIM];

    float *uRaw = new float [XDIM*YDIM];
    float *LuRaw = new float [XDIM*YDIM];
    array_t u = reinterpret_cast<array_t>(*uRaw);
    array_t Lu = reinterpret_cast<array_t>(*LuRaw);

    Timer timer;

    for(int test = 1; test <= 10; test++)
    {
        std::cout << "Running test iteration " << std::setw(2) << test << " ";
        timer.Start();
        ComputeLaplacian(u, Lu);
        timer.Stop("Elapsed time : ");
    }

    return 0;
}
```

This is the actual call to our "benchmark" (executed and measured several times)

Kernel header (Laplacian.h)

LaplacianStencil_0_0

```
#pragma once

#define XDIM 16384
#define YDIM 16384

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM]);
```

Kernel header (Laplacian.h)

LaplacianStencil_0_0

Size of grid presumed constant and known at time of compilation

```
#pragma once

#define XDIM 16384
#define YDIM 16384

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM]);
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_0

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int i = 1; i < XDIM-1; i++)
    for (int j = 1; j < YDIM-1; j++)
        Lu[i][j] =
        -4 * u[i][j]
        + u[i+1][j]
        + u[i-1][j]
        + u[i][j+1]
        + u[i][j-1];

}
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_0

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int i = 1; i < XDIM-1; i++)
    for (int j = 1; j < YDIM-1; j++)
        Lu[i][j] =
            -4 * u[i][j]
            + u[i+1][j]
            + u[i-1][j]
            + u[i][j+1]
            + u[i][j-1];

}
```

OpenMP used to parallelize outer loop

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_0

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int i = 1; i < XDIM-1; i++)
    for (int j = 1; j < YDIM-1; j++)
        Lu[i][j] =
        -4 * u[i][j]
        + u[i+1][j]
        + u[i-1][j]
        + u[i][j+1]
        + u[i][j-1];

}
```

Execution:

```
Running test iteration  1 [Elapsed time : 120.472ms]
Running test iteration  2 [Elapsed time : 25.3752ms]
Running test iteration  3 [Elapsed time : 24.3025ms]
Running test iteration  4 [Elapsed time : 23.0271ms]
Running test iteration  5 [Elapsed time : 22.6208ms]
Running test iteration  6 [Elapsed time : 22.9576ms]
Running test iteration  7 [Elapsed time : 22.5305ms]
Running test iteration  8 [Elapsed time : 23.7184ms]
Running test iteration  9 [Elapsed time : 22.691ms]
Running test iteration 10 [Elapsed time : 24.7188ms]
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_1

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

    for (int i = 1; i < XDIM-1; i++)
    for (int j = 1; j < YDIM-1; j++)
        Lu[i][j] =
            -4 * u[i][j]
            + u[i+1][j]
            + u[i-1][j]
            + u[i][j+1]
            + u[i][j-1];

}
```

Without OpenMP parallelization

Execution:

```
Running test iteration  1 [Elapsed time : 678.226ms]
Running test iteration  2 [Elapsed time : 244.218ms]
Running test iteration  3 [Elapsed time : 244.315ms]
Running test iteration  4 [Elapsed time : 246.056ms]
Running test iteration  5 [Elapsed time : 244.506ms]
Running test iteration  6 [Elapsed time : 243.8ms]
Running test iteration  7 [Elapsed time : 243.287ms]
Running test iteration  8 [Elapsed time : 245.844ms]
Running test iteration  9 [Elapsed time : 244.315ms]
Running test iteration 10 [Elapsed time : 245.566ms]
```

Kernel header (Laplacian.h)

LaplacianStencil_0_2

Size reduced 16K -> 4K

```
#pragma once

#define XDIM 4096
#define YDIM 4096

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM]);
```

Execution:

```
Running test iteration  1 [Elapsed time : 21.3287ms]
Running test iteration  2 [Elapsed time : 2.81527ms]
Running test iteration  3 [Elapsed time : 1.66752ms]
Running test iteration  4 [Elapsed time : 1.57543ms]
Running test iteration  5 [Elapsed time : 1.50367ms]
Running test iteration  6 [Elapsed time : 1.48125ms]
Running test iteration  7 [Elapsed time : 1.52556ms]
Running test iteration  8 [Elapsed time : 1.33879ms]
Running test iteration  9 [Elapsed time : 1.3976ms]
Running test iteration 10 [Elapsed time : 1.40909ms]
```

Kernel header (Laplacian.h)

LaplacianStencil_0_3

Size reduced 16K -> 2K

```
#pragma once

#define XDIM 2048
#define YDIM 2048

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM]);
```

Execution:

```
Running test iteration  1 [Elapsed time : 23.4823ms]
Running test iteration  2 [Elapsed time : 9.35612ms]
Running test iteration  3 [Elapsed time : 3.60061ms]
Running test iteration  4 [Elapsed time : 7.08704ms]
Running test iteration  5 [Elapsed time : 0.438221ms]
Running test iteration  6 [Elapsed time : 8.44043ms]
Running test iteration  7 [Elapsed time : 4.80748ms]
Running test iteration  8 [Elapsed time : 6.9574ms]
Running test iteration  9 [Elapsed time : 8.16184ms]
Running test iteration 10 [Elapsed time : 0.285378ms]
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_4

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int j = 1; j < YDIM-1; j++)
    for (int i = 1; i < XDIM-1; i++)
         Lu[i][j] =
         -4 * u[i][j]
         + u[i+1][j]
         + u[i-1][j]
         + u[i][j+1]
         + u[i][j-1];

}
```

Size reduced 16K -> 4K
Loop Order Swapped

Execution:

```
Running test iteration  1 [Elapsed time : 88.9032ms]
Running test iteration  2 [Elapsed time : 50.2971ms]
Running test iteration  3 [Elapsed time : 50.5499ms]
Running test iteration  4 [Elapsed time : 50.2705ms]
Running test iteration  5 [Elapsed time : 51.0571ms]
Running test iteration  6 [Elapsed time : 51.5478ms]
Running test iteration  7 [Elapsed time : 51.4321ms]
Running test iteration  8 [Elapsed time : 50.3991ms]
Running test iteration  9 [Elapsed time : 50.4688ms]
Running test iteration 10 [Elapsed time : 52.8201ms]
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_5

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int j = 1; j < YDIM-1; j++)
    for (int i = 1; i < XDIM-1; i++)
         Lu[i][j] =
         -4 * u[i][j]
         + u[i+1][j]
         + u[i-1][j]
         + u[i][j+1]
         + u[i][j-1];

}
```

Size reduced 16K -> 2K
Loop Order Swapped

Execution:

```
Running test iteration  1 [Elapsed time : 53.1412ms]
Running test iteration  2 [Elapsed time : 2.73531ms]
Running test iteration  3 [Elapsed time : 2.6788ms]
Running test iteration  4 [Elapsed time : 2.66177ms]
Running test iteration  5 [Elapsed time : 2.66733ms]
Running test iteration  6 [Elapsed time : 2.6668ms]
Running test iteration  7 [Elapsed time : 2.63204ms]
Running test iteration  8 [Elapsed time : 2.67448ms]
Running test iteration  9 [Elapsed time : 2.6665ms]
Running test iteration 10 [Elapsed time : 2.66042ms]
```

Kernel Body (Laplacian.cpp)

LaplacianStencil_0_6

```
#include "Laplacian.h"

void ComputeLaplacian(const float (&u)[XDIM][YDIM], float (&Lu)[XDIM][YDIM])
{

#pragma omp parallel for
    for (int j = 1; j < YDIM-1; j++)
    for (int i = 1; i < XDIM-1; i++)
         Lu[i][j] =
         -4 * u[i][j]
         + u[i+1][j]
         + u[i-1][j]
         + u[i][j+1]
         + u[i][j-1];

}
```

Original Size
Loop Order Swapped

Execution:

```
Running test iteration  1 [Elapsed time : 2034.53ms]
Running test iteration  2 [Elapsed time : 1814.3ms]
Running test iteration  3 [Elapsed time : 1873.85ms]
Running test iteration  4 [Elapsed time : 1779.44ms]
Running test iteration  5 [Elapsed time : 1731.12ms]
Running test iteration  6 [Elapsed time : 1809.28ms]
Running test iteration  7 [Elapsed time : 1825.35ms]
Running test iteration  8 [Elapsed time : 1725.44ms]
Running test iteration  9 [Elapsed time : 1806.62ms]
Running test iteration 10 [Elapsed time : 1882.4ms]
```

Benchmark launcher (main.cpp)

LaplacianStencil_0_[0-6]

```
#include "Timer.h"
#include "Laplacian.h"

#include <iomanip>

int main(int argc, char *argv[])
{
    using array_t = float (&) [XDIM][YDIM];

    float *uRaw = new float [XDIM*YDIM];
    float *LuRaw = new float [XDIM*YDIM];
    array_t u = reinterpret_cast<array_t>(*uRaw);
    array_t Lu = reinterpret_cast<array_t>(*LuRaw);

    Timer timer;

    for(int test = 1; test <= 10; test++)
    {
        std::cout << "Running test iteration " << std::setw(2) << test << " ";
        timer.Start();
        ComputeLaplacian(u, Lu);
        timer.Stop("Elapsed time : ");
    }

    return 0;
}
```