

Dense Lower Triangular Solver

Code Hierarchy

File	Functions	Linked Files
Support.h Contain various supporting function for main.cu	<ol style="list-style-type: none">1. verifyResults Take calculated matrix array and compare it with actual matrix to verify correctness.2. printCSV Take matrix array and print it on console3. writeCSV Takes a matrix array and write it in CSV format on file directory system4. loadCSV Take matrix array with reference and csv file, and load csv file data into the matrix array and return	<ol style="list-style-type: none">1. Support.cu (Implementation file)2. Main.cu (Utility)
Kernel.h Contain actual kernel codes to be executed on GPU + some host codes	<ol style="list-style-type: none">1. gpu_simple_solver_kernel Original kernel modified to run correctly2. gpu_simple_solver_Anjum Modified kernel optimized for performance but not for scalability3. gpu_optimized_solver_Anjum Modified kernel optimized for both performance and scalability4. gpu_simple_solver Host code to call appropriate kernel5. Cpu_multiply Host to process multiplication of matrix6. Cpu_solver	<ol style="list-style-type: none">1. Kernel.cu Implementation file2. Main.cu Utility file

	Host to process solve equation using cpu	
main.cu	<p>1. Onhost Called by main program with kernel type parameter and initialize different arrays and forward to the device or gpu</p> <p>2. OnDevice Called by onhost function with host matrix array, initialize variable on device and forward request to gpu or cpu solver.</p>	

Execution Sequence

(Host Code)

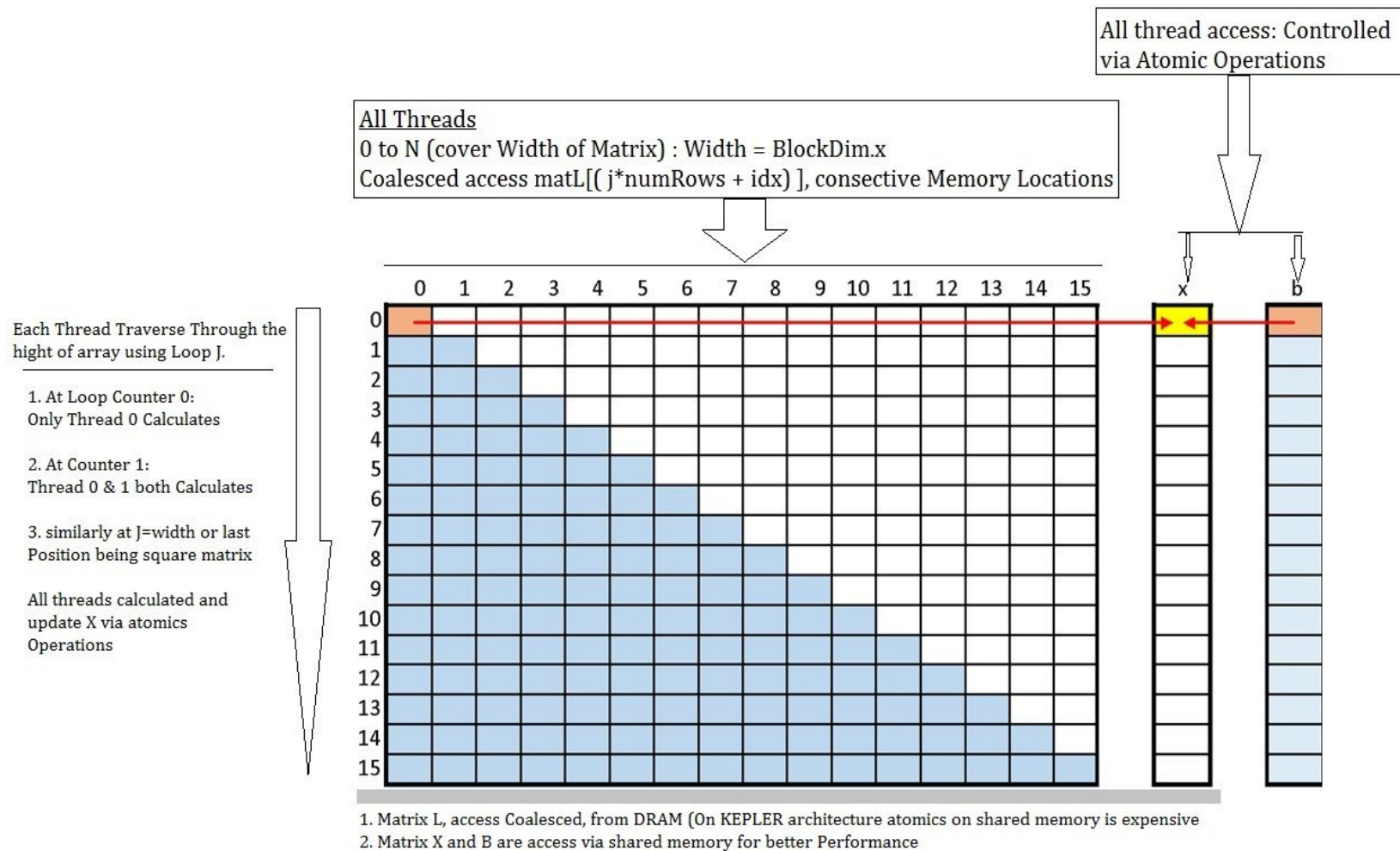
1. On the Command Line call the executable with Parameter or Kernel Type
i.e main 1 or main 5

Kernel Types

- 0 : CPU Solver
 - 1 : Old Simple Solver kernel
 - 2 : gpu_initial solver Anjum
 - 3: gpu Simple solver kernel2
 - 4 : gpu_simple_solver_Anjum
 - 5: gpu_Optimized_Solver_Anjum
2. Main Method call the on host method
 3. OnHost method initializes the input arrays with csv file from file system and call the onDevice method
 4. OnDevice Method initializes the input arrays on device, copy data and call gpu_Solver method with kernel type.
 5. Gpu_solver call the appropriate kernel and return the calculated array back to the ondevice method
 6. Ondevice Method Prints / compare or write the calculated results on csv or display on console

Execution Sequence of KERNEL

(GPU_OPTIMIZED_SOLVER_ANJUM)



```

__global__ void gpu_optimized_solver_Anjum(int* matL, int* vecX, int* vecB, int numRows)
{
    int tot=0;
    int r_matL=0;
    __shared__ int ds_X[N];
    int idx = blockIdx.x*blockDim.x + threadIdx.x;
    if (idx >= numRows)        return;
    ds_X[idx]=0;

    for (int j = 0; j <numRows ; j++)
    {r_matL=matL[(j*numRows + idx) ];
    //__syncthreads();
    if (j> 0 && j>idx)
    {
        tot= (-1 * (r_matL *ds_X[idx]));

        //atomicAdd (&ds_B[j],tot); //ds_B[j]+=tot;    keeper takes time on shared memory for atomics then global memory
        atomicAdd (&vecB[j],tot); //vecB[idx]+=tot;    // Keeper Performs better on atomics on Global Memory then Shared
    }
    else if (idx == j)
    {
        ds_X[j] = vecB[j] / r_matL    ;
    }
    }
    vecX[idx] = ds_X[idx];
}

```

Previous kernel Performance 76.13us for 32x32 matrix

```
C:\Users\ANJUM\Documents\Visual Studio 2015\Projects\Matrix\Matrix>test
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
C:\Users\ANJUM\Documents\Visual Studio 2015\Projects\Matrix\Matrix>nvprof test
==4596== NVPROF is profiling process 4596, command: test
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
==4596== Profiling application: test
==4596== Warning: Found 9 invalid records in the result.
==4596== Warning: This can happen if device ran out of memory or if a device kernel was stopped due
to an assertion.
==4596== Profiling result:
Type      Time(%)      Time      Calls      Avg      Min      Max      Name
GPU activities: 93.81% 76.128us 32 2.3790us 2.3040us 2.8160us gpu_simple_solver_kerne
l(int*, int*, int*, int, int)
    4.14% 3.3600us 2 1.6800us 1.2160us 2.1440us [CUDA memcpy HtoD]
    2.05% 1.6640us 1 1.6640us 1.6640us 1.6640us [CUDA memcpy DtoH]
API calls: 67.22% 164.96ms 3 54.986ms 11.899us 164.30ms cudaMalloc
    31.09% 76.303ms 1 76.303ms 76.303ms 76.303ms cuDevicePrimaryCtxRelea
se
    0.40% 989.70us 32 30.928us 23.097us 228.88us cudaLaunchKernel
    0.40% 977.45us 1 977.45us 977.45us 977.45us cuDeviceGetName
    0.40% 971.15us 1 971.15us 971.15us 971.15us cuModuleUnload
    0.34% 832.56us 87 9.5690us 349ns 424.51us cuDeviceGetAttribute
    0.08% 191.43us 3 63.809us 50.394us 84.691us cudaMemcpy
    0.06% 143.49us 1 143.49us 143.49us 143.49us cuDeviceSynchronize
    0.01% 20.298us 1 20.298us 20.298us 20.298us cuDeviceTotalMem
    0.01% 14.698us 1 14.698us 14.698us 14.698us cuDeviceGetPCIBusId
    0.00% 4.2000us 3 1.4000us 350ns 3.5000us cuDeviceGetCount
    0.00% 4.1990us 2 2.0990us 350ns 3.8490us cuDeviceGet
```

Optimized kernel Performance 33.12us for 32x32 matrix

```
C:\Windows\system32\cmd.exe
to an assertion.
==1740== Profiling result:
Type      Time(%)      Time      Calls      Avg      Min      Max      Name
GPU activities: 86.90% 33.120us 1 33.120us 33.120us 33.120us gpu_optimized_solver_An
jum(int*, int*, int*, int)
    8.65% 3.2960us 2 1.6480us 1.2160us 2.0800us [CUDA memcpy HtoD]
    4.45% 1.6960us 1 1.6960us 1.6960us 1.6960us [CUDA memcpy DtoH]
API calls: 68.80% 162.85ms 3 54.283ms 11.899us 162.20ms cudaMalloc
    29.92% 70.816ms 1 70.816ms 70.816ms 70.816ms cuDevicePrimaryCtxRelea
se
    0.43% 1.0184ms 1 1.0184ms 1.0184ms 1.0184ms cuModuleUnload
    0.32% 765.72us 82 9.3380us 349ns 379.71us cuDeviceGetAttribute
    0.29% 684.53us 1 684.53us 684.53us 684.53us cuDeviceGetName
    0.09% 212.78us 3 70.925us 49.694us 106.74us cudaMemcpy
    0.09% 205.08us 1 205.08us 205.08us 205.08us cudaLaunchKernel
    0.04% 93.091us 1 93.091us 93.091us 93.091us cuDeviceSynchronize
    0.01% 20.647us 1 20.647us 20.647us 20.647us cuDeviceTotalMem
    0.01% 16.098us 1 16.098us 16.098us 16.098us cuDeviceGetPCIBusId
    0.00% 5.5990us 3 1.8660us 700ns 4.1990us cuDeviceGetCount
    0.00% 3.5000us 2 1.7500us 350ns 3.1500us cuDeviceGet
```