Walchand College of Engineering, Sangli

Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2022-23 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 9

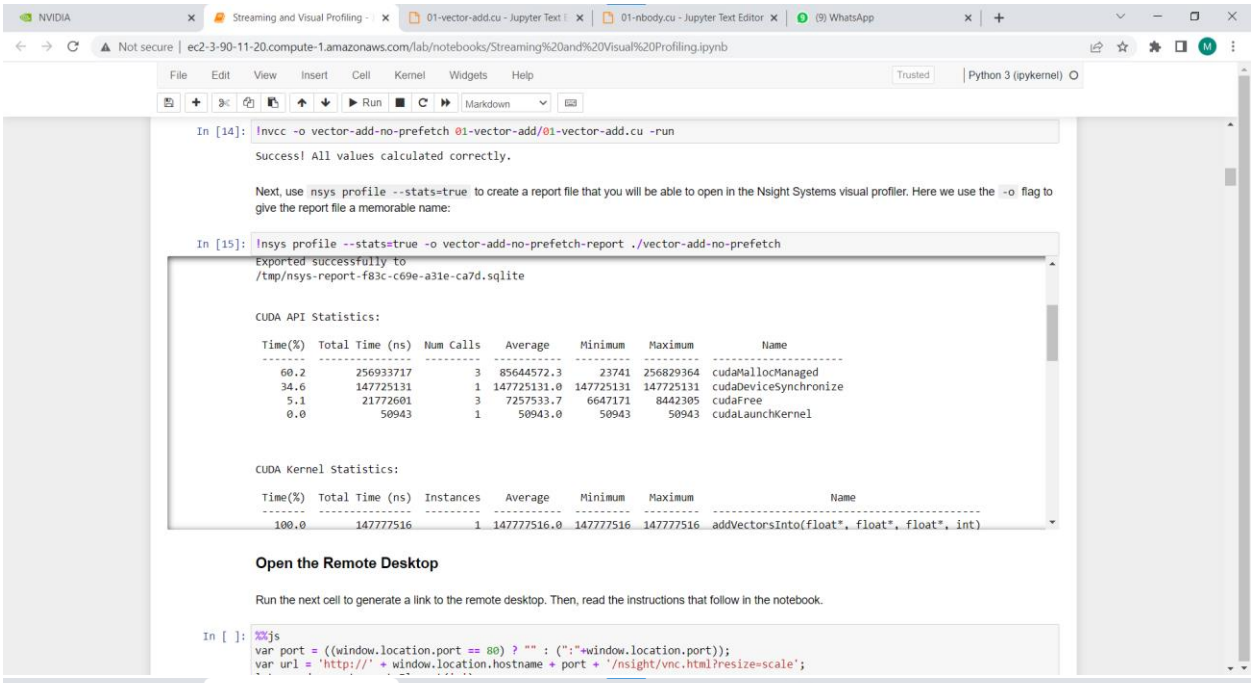
Exam Seat No: 2019BTECS00064

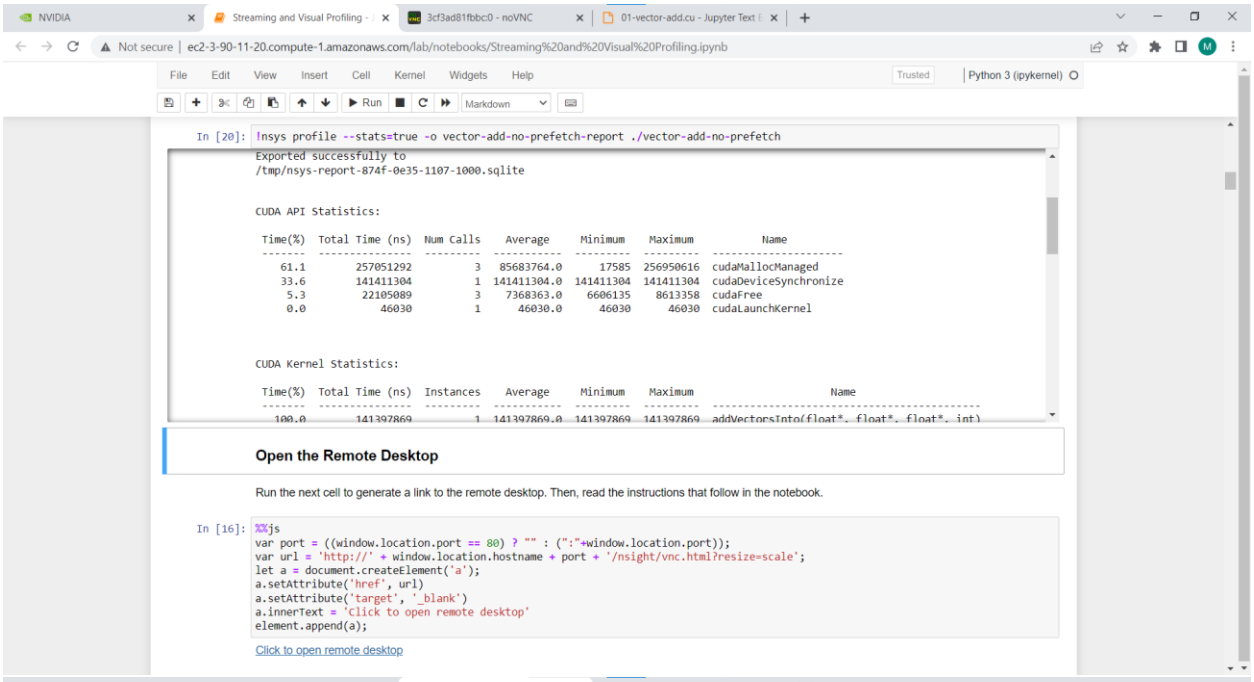
Name – Kunal Santosh Kadam

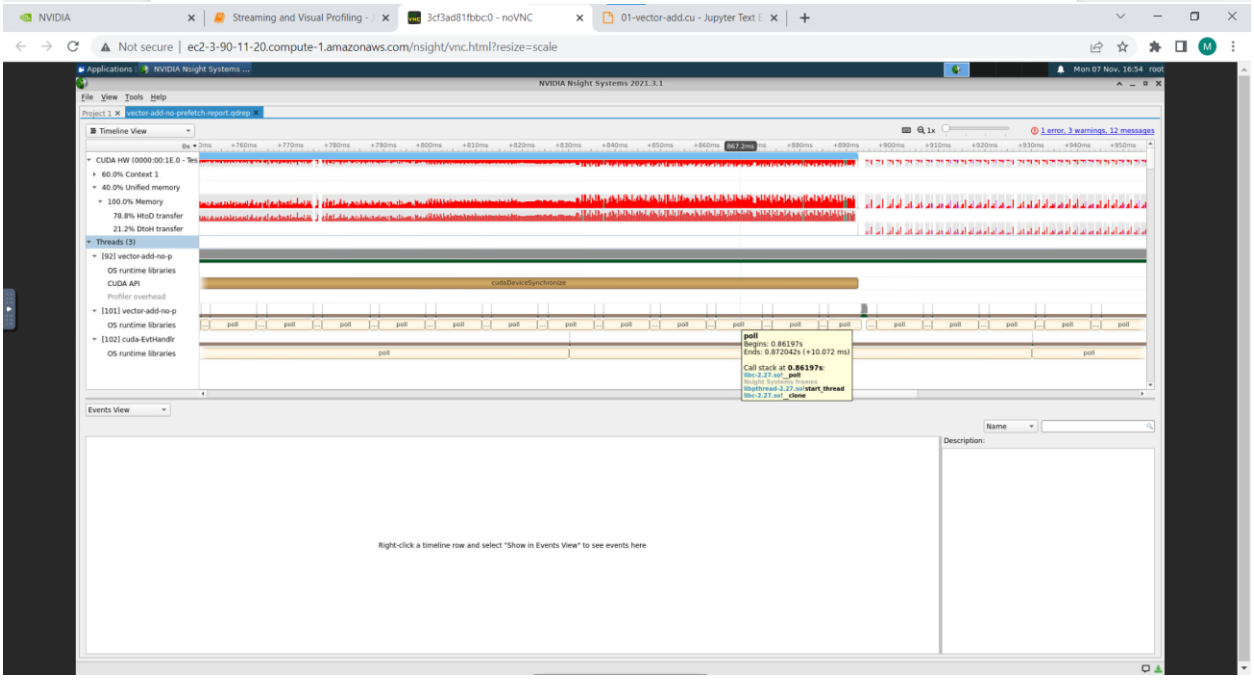
Problem Statement 1:

Implement Vector-Vector addition using CUDA C. State and justify the speedup using different size of threads and blocks.

Screenshot #:




NVIDIA
Streaming and Visual Profiling -
3cf3ad81fbbc0 - noVNC
01-vector-add.cu - Jupyter Text
Not secure | ec2-3-90-11-20.compute-1.amazonaws.com/lab/notebooks/Streaming%20and%20Visual%20Profiling.ipynb
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
In [20]: !nsys profile --stats=true -o vector-add-no-prefetch-report ./vector-add-no-prefetch
Exported successfully to
/tmp/nsys-report-874f-0e35-1107-1000.sqlite
CUDA API Statistics:
Time(%) Total Time (ns) Num Calls Average Minimum Maximum Name
61.1 257051292 3 85683764.0 17585 256950616 cudaMallocManaged
33.6 141411304 1 141411304.0 141411304 141411304 cudaDeviceSynchronize
5.3 22105089 3 7368363.0 6606135 8613358 cudaFree
0.0 46030 1 46030.0 46030 46030 cudaLaunchKernel
CUDA Kernel Statistics:
Time(%) Total Time (ns) Instances Average Minimum Maximum Name
100.0 141397869 1 141397869.0 141397869 141397869 addVectorsInto(float*, float*, float*, int)
Open the Remote Desktop
Run the next cell to generate a link to the remote desktop. Then, read the instructions that follow in the notebook.
In [16]: %%js
var port = ((window.location.port == 80) ? "" : (":"+window.location.port));
var url = 'http://' + window.location.hostname + port + '/nsight/vnc.html?resize=scale';
let a = document.createElement('a');
a.setAttribute('href', url)
a.setAttribute('target', '_blank')
a.innerText = 'Click to open remote desktop'
element.append(a);
Click to open remote desktop


NVIDIA
Streaming and Visual Profiling -
3cf3ad81fbbc0 - noVNC
01-vector-add.cu - Jupyter Text
Not secure | ec2-3-90-11-20.compute-1.amazonaws.com/nsight/vnc.html?resize=scale
NVIDIA Nsight Systems 2021.3.1
Timeline View
Events View
Right-click a timeline row and select "Show in Events View" to see events here

Information #:

```
#include <stdio.h>

void initWith(float num, float *a, int N)
{
	for(int i = 0; i < N; ++i)
	{
		a[i] = num;
	}
}

__global__ void addVectorsInto(float *result, float *a, float *b, int N)
{
	int index = threadIdx.x + blockIdx.x * blockDim.x;
	int stride = blockDim.x * gridDim.x;

	for(int i = index; i < N; i += stride)
	{
		result[i] = a[i] + b[i];
	}
}

void checkElementsAre(float target, float *vector, int N)
{
	for(int i = 0; i < N; i++)
	{
		if(vector[i] != target)
		{
			printf("FAIL: vector[%d] - %0.0f does not equal %0.0f\n",
i, vector[i], target);
			exit(1);
		}
	}
```

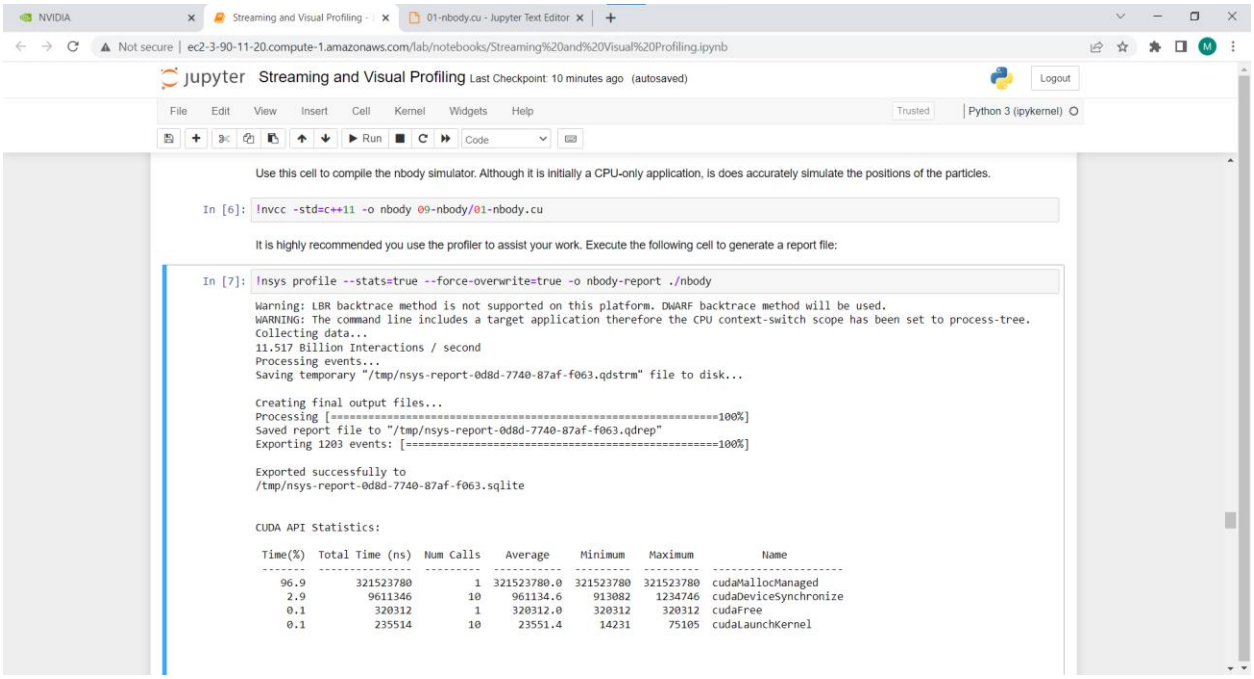
```
        printf("Success! All values calculated correctly.\n");
}

int main()
{
        int deviceId;
        int numberOfSMs;

        cudaGetDevice(&deviceId);
        cudaDeviceGetAttribute(&numberOfSMs,
cudaDevAttrMultiProcessorCount, deviceId);

        const int N = 2<<24;
        size_t size = N * sizeof(float);

        float *a;
        float *b;
        float *c;

        cudaMallocManaged(&a, size);
        cudaMallocManaged(&b, size);
        cudaMallocManaged(&c, size);

        initWith(3, a, N);
        initWith(4, b, N);
        initWith(0, c, N);

        size_t threadsPerBlock;
        size_t numberOfBlocks;

        threadsPerBlock = 256;
        numberOfBlocks = 32 * numberOfSMs;

        cudaError_t addVectorsErr;
```

```
	cudaError_t asyncErr;

	addVectorsInto<<<numberOfBlocks, threadsPerBlock>>>(c, a, b, N);

	addVectorsErr = cudaGetLastError();
	if(addVectorsErr != cudaSuccess) printf("Error: %s\n",
cudaGetErrorString(addVectorsErr));

	asyncErr = cudaDeviceSynchronize();
	if(asyncErr != cudaSuccess) printf("Error: %s\n",
cudaGetErrorString(asyncErr));

	checkElementsAre(7, c, N);

	cudaFree(a);
	cudaFree(b);
	cudaFree(c);
}
```

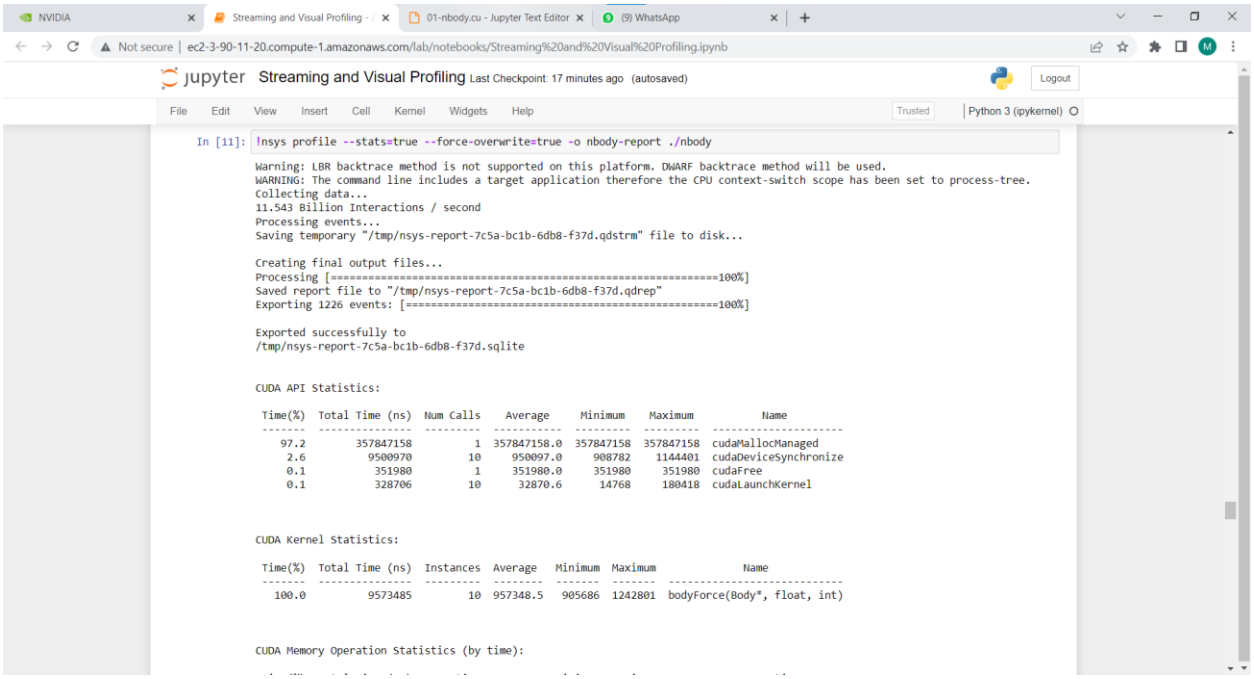
Problem Statement 2:

Implement N-Body Simulator using CUDA C. State and justify the speedup using different size of threads and blocks.

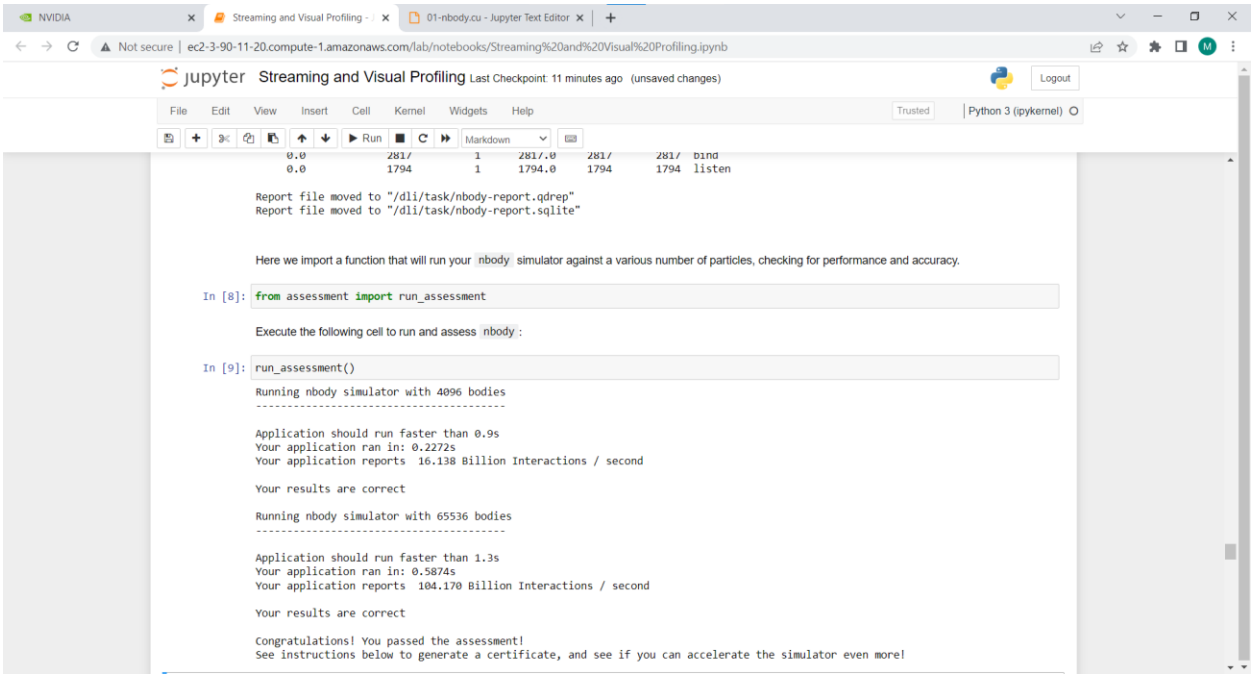
Screenshot #:



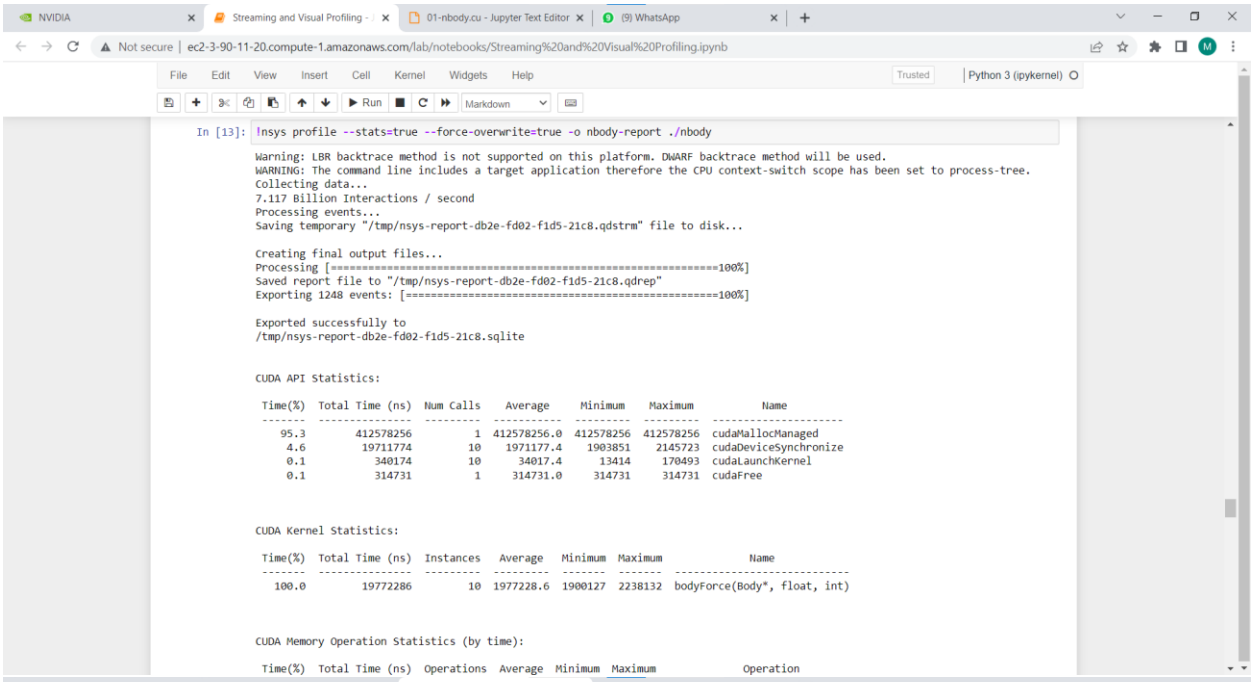
Number of thread per block – 32 with block size 1024

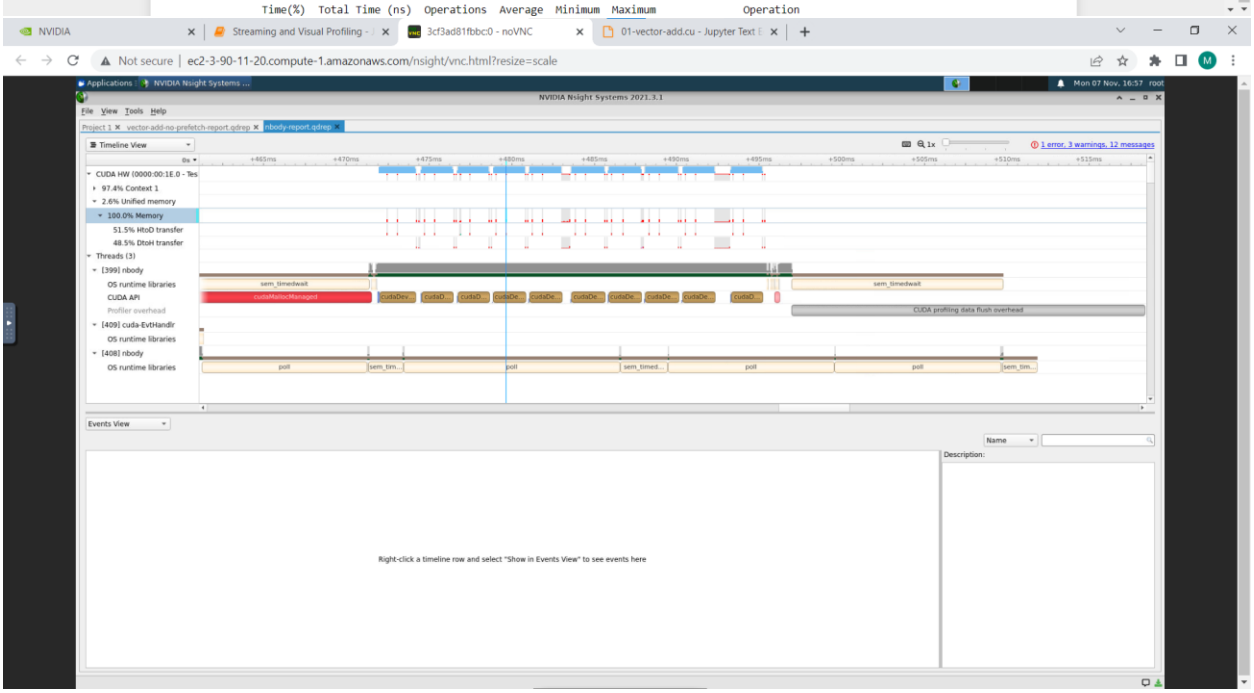


Number of thread per block – 32 with block size 512



Number of thread per block – 4 with block size 1024





Information #:

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "timer.h"
#include "files.h"

#define SOFTENING 1e-9f

typedef struct { float x, y, z, vx, vy, vz; } Body;

__global__ void bodyForce(Body *p, float dt, int n)
{
	int index = threadIdx.x + blockIdx.x * blockDim.x;
	int stride = blockDim.x * gridDim.x;

	for(int i = index; i < N; i += stride)
	{
		float Fx = 0.0f; float Fy = 0.0f; float Fz = 0.0f;

		for (int j = 0; j < n; j++)
		{
			float dx = p[j].x - p[i].x;
			float dy = p[j].y - p[i].y;
			float dz = p[j].z - p[i].z;
			float distSqr = dx*dx + dy*dy + dz*dz + SOFTENING;
			float invDist = rsqrtf(distSqr);
			float invDist3 = invDist * invDist * invDist;

			Fx += dx * invDist3; Fy += dy * invDist3; Fz += dz *
invDist3;
		}
```

```
            p[i].vx += dt*Fx; p[i].vy += dt*Fy; p[i].vz += dt*Fz;
        }
}


int main(const int argc, const char** argv)
{
        // The assessment will test against both 2<11 and 2<15.
        // Feel free to pass the command line argument 15 when you
generate ./nbody report files
        y report files
        int nBodies = 2<<11;
        if (argc > 1) nBodies = 2<<atoi(argv[1]);

        // The assessment will pass hidden initialized values to check for
correctness.
        // You should not make changes to these files, or else the assessment
will not work.
        const char * initialized_values;
        const char * solution_values;

        if (nBodies == 2<<11)
        {
                initialized_values = "09-nbody/files/initialized_4096";
                solution_values = "09-nbody/files/solution_4096";
        }
        else
        { // nBodies == 2<<15
                initialized_values = "09-nbody/files/initialized_65536";
                solution_values = "09-nbody/files/solution_65536";
        }

        if (argc > 2)
                initialized_values = argv[2];
```

```
if (argc > 3)
        solution_values = argv[3];

const float dt = 0.01f; // Time step
const int nIters = 10;  // Simulation iterations

int bytes = nBodies * sizeof(Body);
float *buf;

buf = (float *)malloc(bytes);
cudaMallocManaged(&buf, bytes)

Body *p = (Body*)buf;

read_values_from_file(initialized_values, buf, bytes);

double totalTime = 0.0;

for (int iter = 0; iter < nIters; iter++) {
        StartTimer();

        bodyForce<<<1024,32>>>(p, dt, nBodies); // compute
interbody forces
        cudaDeviceSynchronize();

        for (int i = 0 ; i < nBodies; i++)
        { // integrate position
                p[i].x += p[i].vx*dt;
                p[i].y += p[i].vy*dt;
                p[i].z += p[i].vz*dt;
        }

        const double tElapsed = GetTimer() / 1000.0;
        totalTime += tElapsed;
```

```
        }

        double avgTime = totalTime / (double)(nIters);
        float billionsOfOpsPerSecond = 1e-9 * nBodies * nBodies / avgTime;
        write_values_to_file(solution_values, buf, bytes);

        printf("%0.3f Billion Interactions / second\n",
billionsOfOpsPerSecond);

        cudaFree(buf);
}
```

Github Link:

https://github.com/Kunalkadam179/HPC-Assignment/tree/main/Assignment%20-%209