Class: Final Year (Computer Science and Engineering)

Year: 2021-22 **Semester:** 1 **Course:** High Performance Computing Lab

PRN: 2018BTECS00049

Practical No. 8

Problem Statement 1: Write a CUDA C program to perform the simple matrix-matrix multiplication. Perform code optimization and profiling of existing CUDA C program. (Attach Snapshot of execution before optimization and after optimization)

Ans:

Simple CUDA program

```
%%cu
#include<bits/stdc++.h>
using namespace std;
const int r1 = 20, c1 = 30, r2 = 30, c2 = 20;
__global__ void mulMatrix(int *a, int * b, int *c)
{
    int x = blockIdx.x;
    int y = blockIdx.y;

    int id_c = c2 * y + x;

    c[id_c] = 0;
    for(int k = 0 ; k < c1; k++)
    {
        int id_a = c1 * y + k;
        int id_b = c2 * k + x;
        c[id_c] = c[id_c] + a[id_a] * b[id_b];
    }
}
```

```
int main()
{
    int host_a[r1][c1] , host_b[r2][c2], host_c[r1][c2];
    int *devc_a, *devc_b, *devc_c;

    cudaMalloc(&devc_a, r1 * c1 * sizeof(int));
    cudaMalloc(&devc_b, r2 * c2 * sizeof(int));
    cudaMalloc(&devc_c, r1 * c2 * sizeof(int));

    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c1;j++)
          host_a[i][j] = i + j + 1;
    }

    for(int i=0;i<r2;i++)
    {
        for(int j=0;j<c2;j++)
          host_b[i][j] = (i + 1) * (j + 1);
    }
```

```
    cudaMemcpy(devc_a, host_a, r1 * c1 * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(devc_b, host_b, r2 * c2 * sizeof(int), cudaMemcpyHostToDevice);

    dim3 grid(c2,r1);

    mulMatrix<<<grid, 1>>>(devc_a, devc_b, devc_c);

    cudaMemcpy(host_c, devc_c, r1 * c2 * sizeof(int), cudaMemcpyDeviceToHost);

    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c2;j++)
          cout<<host_c[i][j]<<" ";
        cout<<endl;
    }
    cudaFree(devc_a);
    cudaFree(devc_b);
    cudaFree(devc_c);
    return 0;
}
```

```
}
```

```
9455 18910 28365 37820 47275 56730 66185 75640 85095 94550 104005 113460 122915 132370 141825 151280 160735 170190 179645 189100
9920 19840 29760 39680 49600 59520 69440 79360 89280 99200 109120 119040 128960 138880 148800 158720 168640 178560 188480 198400
10385 20770 31155 41540 51925 62310 72695 83080 93465 103850 114235 124620 135005 145390 155775 166160 176545 186930 197315 207700
10850 21700 32550 43400 54250 65100 75950 86800 97650 108500 119350 130200 141050 151900 162750 173600 184450 195300 206150 217000
11315 22630 33945 45260 56575 67890 79205 90520 101835 113150 124465 135780 147095 158410 169725 181040 192355 203670 214985 226300
11780 23560 35340 47120 58900 70680 82460 94240 106020 117800 129580 141360 153140 164920 176700 188480 200260 212040 223820 235600
12245 24490 36735 48980 61225 73470 85715 97960 110205 122450 134695 146940 159185 171430 183675 195920 208165 220410 232655 244900
12710 25420 38130 50840 63550 76260 88970 101680 114390 127100 139810 152520 165230 177940 190650 203360 216070 228780 241490 254200
13175 26350 39525 52700 65875 79050 92225 105400 118575 131750 144925 158100 171275 184450 197625 210800 223975 237150 250325 263500
13640 27280 40920 54560 68200 81840 95480 109120 122760 136400 150040 163680 177320 190960 204600 218240 231880 245520 259160 272800
14105 28210 42315 56420 70525 84630 98735 112840 126945 141050 155155 169260 183365 197470 211575 225680 239785 253890 267995 282100
14570 29140 43710 58280 72850 87420 101990 116560 131130 145700 160270 174840 189410 203980 218550 233120 247690 262260 276830 291400
15035 30070 45105 60140 75175 90210 105245 120280 135315 150350 165385 180420 195455 210490 225525 240560 255595 270630 285665 300700
15500 31000 46500 62000 77500 93000 108500 124000 139500 155000 170500 186000 201500 217000 232500 248000 263500 279000 294500 310000
15965 31930 47895 63860 79825 95790 111755 127720 143685 159650 175615 191580 207545 223510 239475 255440 271405 287370 303335 319300
16430 32860 49290 65720 82150 98580 115010 131440 147870 164300 180730 197160 213590 230020 246450 262880 279310 295740 312170 328600
16895 33790 50685 67580 84475 101370 118265 135160 152055 168950 185845 202740 219635 236530 253425 270320 287215 304110 321005 337900
17360 34720 52080 69440 86800 104160 121520 138880 156240 173600 190960 208320 225680 243040 260400 277760 295120 312480 329840 347200
17825 35650 53475 71300 89125 106950 124775 142600 160425 178250 196075 213900 231725 249550 267375 285200 303025 320850 338675 356500
18290 36580 54870 73160 91450 109740 128030 146320 164610 182900 201190 219480 237770 256060 274350 292640 310930 329220 347510 365800
```

Problem Statement 2: Write a CUDA C program to demonstrate the use of different GPU memories.

- **Use of private memory.**
- **Use of shared memory.**
- **Use of global memory**

Optimized code using shared memory

6s

```
%%cu
#include<bits/stdc++.h>
using namespace std;

const int r1 = 20, c1 = 30, r2 = 30, c2 = 20;

__global__ void mulMatrix(int *a, int * b, int *c)
{
    int x = blockIdx.x;
    int y = blockIdx.y;
    int k = threadIdx.x;

    __shared__ int p[c1];

    int id_c = c2 * y + x;
    c[id_c] = 0;
    int id_a = c1 * y + k;
    int id_b = c2 * k + x;

    p[k] = a[id_a] * b[id_b];
```

```
     __syncthreads();

    for(int i=0;i<c1;i++)
      c[id_c] = c[id_c] + p[i];

}


int main()
{
    int host_a[r1][c1] , host_b[r2][c2], host_c[r1][c2];
    int *devc_a, *devc_b, *devc_c;

    cudaMalloc(&devc_a, r1 * c1 * sizeof(int));
    cudaMalloc(&devc_b, r2 * c2 * sizeof(int));
    cudaMalloc(&devc_c, r1 * c2 * sizeof(int));

    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c1;j++)
          host_a[i][j] = i + j + 1;
    }
```

```
    for(int i=0;i<r2;i++)
    {
        for(int j=0;j<c2;j++)
          host_b[i][j] = (i + 1) * (j + 1);
    }

    cudaMemcpy(devc_a, host_a, r1 * c1 * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(devc_b, host_b, r2 * c2 * sizeof(int), cudaMemcpyHostToDevice);

    dim3 grid(c2,r1);

    mulMatrix<<<grid,c1>>>(devc_a, devc_b, devc_c);

    cudaMemcpy(host_c, devc_c, r1 * c2 * sizeof(int), cudaMemcpyDeviceToHost);

    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c2;j++)
          cout<<host_c[i][j]<<" ";
        cout<<endl;
    }
    cudaFree(devc_a);
```

```
    cudaFree(devc_b);
    cudaFree(devc_c);
    return 0;
}
```

```
9455 18910 28365 37820 47275 56730 66185 75640 85095 94550 104005 113460 122915 132370 141825 151280 160735 170190 179645 189100
9920 19840 29760 39680 49600 59520 69440 79360 89280 99200 109120 119040 128960 138880 148800 158720 168640 178560 188480 198400
10385 20770 31155 41540 51925 62310 72695 83080 93465 103850 114235 124620 135005 145390 155775 166160 176545 186930 197315 207700
10850 21700 32550 43400 54250 65100 75950 86800 97650 108500 119350 130200 141050 151900 162750 173600 184450 195300 206150 217000
11315 22630 33945 45260 56575 67890 79205 90520 101835 113150 124465 135780 147095 158410 169725 181040 192355 203670 214985 226300
11780 23560 35340 47120 58900 70680 82460 94240 106020 117800 129580 141360 153140 164920 176700 188480 200260 212040 223820 235600
12245 24490 36735 48980 61225 73470 85715 97960 110205 122450 134695 146940 159185 171430 183675 195920 208165 220410 232655 244900
12710 25420 38130 50840 63550 76260 88970 101680 114390 127100 139810 152520 165230 177940 190650 203360 216070 228780 241490 254200
13175 26350 39525 52700 65875 79050 92225 105400 118575 131750 144925 158100 171275 184450 197625 210800 223975 237150 250325 263500
13640 27280 40920 54560 68200 81840 95480 109120 122760 136400 150040 163680 177320 190960 204600 218240 231880 245520 259160 272800
14105 28210 42315 56420 70525 84630 98735 112840 126945 141050 155155 169260 183365 197470 211575 225680 239785 253890 267995 282100
14570 29140 43710 58280 72850 87420 101990 116560 131130 145700 160270 174840 189410 203980 218550 233120 247690 262260 276830 291400
15035 30070 45105 60140 75175 90210 105245 120280 135315 150350 165385 180420 195455 210490 225525 240560 255595 270630 285665 300700
15500 31000 46500 62000 77500 93000 108500 124000 139500 155000 170500 186000 201500 217000 232500 248000 263500 279000 294500 310000
15965 31930 47895 63860 79825 95790 111755 127720 143685 159650 175615 191580 207545 223510 239475 255440 271405 287370 303335 319300
16430 32860 49290 65720 82150 98580 115010 131440 147870 164300 180730 197160 213590 230020 246450 262880 279310 295740 312170 328600
16895 33790 50685 67580 84475 101370 118265 135160 152055 168950 185845 202740 219635 236530 253425 270320 287215 304110 321005 337900
17360 34720 52080 69440 86800 104160 121520 138880 156240 173600 190960 208320 225680 243040 260400 277760 295120 312480 329840 347200
17825 35650 53475 71300 89125 106950 124775 142600 160425 178250 196075 213900 231725 249550 267375 285200 303025 320850 338675 356500
18290 36580 54870 73160 91450 109740 128030 146320 164610 182900 201190 219480 237770 256060 274350 292640 310930 329220 347510 365800
```