IS2202-Laboration 2
Simulating Graphics Processing Units (GPUs) in GPGPU-Sim

YiHang Chen: yihche@ug.kth.se
Ruolan Wang :ruolanw@kth.se

Question 1: What is the CUDA kernel called, and what is its function signature? What is the datatype that we used in the matrix multiplication?

he CUDA kernel responsible for matrix multiplication is named `matrix_multiply`. This kernel is invoked using the `<<<...>>>` syntax .

```
matrix_multiply<<<dim_grid, dim_block>>>(A_gpu, B_gpu, C_gpu, N);
```

`matrix_multiply` is the name of the CUDA kernel function. It takes four arguments: pointers to the matrices `A_gpu`, `B_gpu`, and `C_gpu`, as well as the size of the matrices `N`. This kernel performs the actual computation of matrix multiplication on the GPU.

The function signature is:
__global__ void matrix_multiply(float *A, float *B, float* C, int N)
`__global__`: This keyword specifies that the function is a CUDA kernel, meaning it will be executed on the GPU.

`float *A`: Pointer to the first matrix `A`.

`float *B`: Pointer to the second matrix `B`.

`float *C`: Pointer to the result matrix `C`.

`int N`: Size of the matrices (assuming they are square matrices of size NxN).

Datatype:float

Question 2: Analyze the GEMM implementation and derive a formula for how many Floating-Point Operations (FLOP) are needed to compute a N x N matrix multiplication (hint: look at the simpler naive CPU-version called matrix_multiply_cpu).

In the `matrix_multiply_cpu` function, there are three nested loops iterating over the dimensions of the matrices A, B, and C. The innermost loop performs a single floating-point multiplication and addition operation.

Here's a breakdown of the FLOPs:

Inside the innermost loop, there is one floating-point multiplication (`A[i * N + k] * B[k * N + j]`) and one floating-point addition (`tmp += ...`).

So, for each element of the resulting matrix C, there are N multiplication and (N-1) addition, which is 2*N-1 in total

Since there are N*N elements in matrix C, the total number of FLOPs for the entire matrix multiplication operation is: (2*N-1)*N*N.

Question 3: Look up the NVIDIA GTX480 GPU and describe its characteristics. What year was it introduced? What was its peak performance and external memory bandwidth? What GPU architecture family did it belong to? What is the GPU core frequency?

The NVIDIA GTX 480 GPU was introduced in March 2010 as part of NVIDIA's GeForce 400 series.

Peak Performance: The GTX 480 had a peak single-precision floating-point performance of approximately 1.35 teraflops (trillion floating-point operations per second),Pixel Rate 21.03 GPixel/s ,Texture Rate 42.06 GTexel/s .

External Memory Bandwidth: The GTX 480 featured a memory bandwidth of 177.4 GB/s. This was achieved using a 384-bit memory interface with GDDR5 memory.

GPU Architecture Family: The GTX 480 belonged to the Fermi architecture family. Fermi was NVIDIA's first architecture to introduce features like CUDA cores (stream processors), Tessellation units, and concurrent kernel execution.

GPU Core Frequency: The core clock frequency of the GTX 480 was typically around 700 MHz, though actual frequencies could vary depending on factors like thermal conditions and manufacturer-specific overclocking.

Question 4: How long time did the simulation take? What was the estimate slowdown of the simulator ("…"). Why is the slowdown so large compared to executing on a real GPU?
gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 1 sec (1 sec)`
gpgpu_simulation_rate = 154624 (inst/sec)
gpgpu_simulation_rate = 5368 (cycle/sec)
gpgpu_silicon_slowdown = 130402x
GPGPU-Sim: synchronize waiting for inactive GPU simulation
GPGPU-Sim API: Stream Manager State
GPGPU-Sim: detected inactive GPU simulation thread

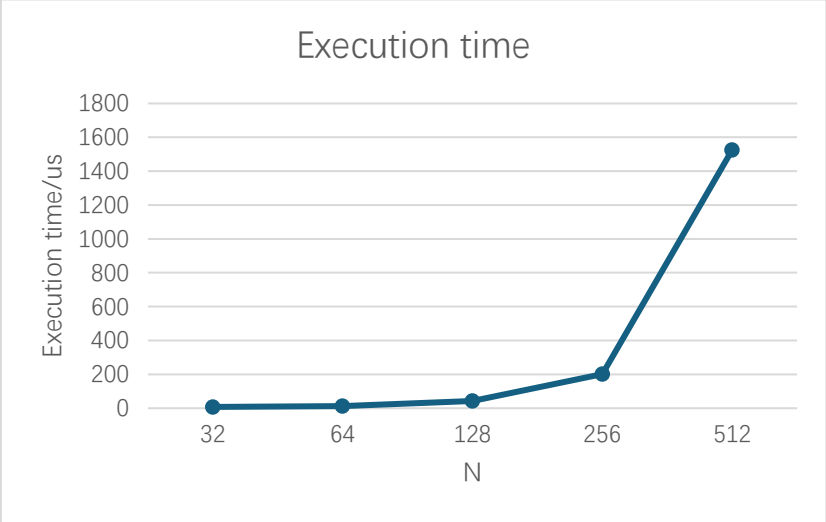
The simulation takes 1 second. The slowdown of the simulator is 130402 times slower than the real execution time.
While simulation environments can provide valuable insights and debugging capabilities, they often cannot match the performance of executing code directly on real GPU hardware due to the overhead and limitations inherent in the simulation process.

Question 5: Analyze stat.txt and answer the following questions:
a) How many clock cycles did it take to execute the GEMM kernel?
b) How many instructions were executed?
c) How high IPC (instructions-per-cycle) did the GPU achieve?
d) DRAM Bandwidth obtained and what L2 cache hit rate was achieved?
e) What performance in terms of #FLOPS/s did we achieve
hint: use information from exercise 1 to calculate the FLOP/s.
a) gpu_sim_cycle = 5368 // it takes 5368 clock cycles to execute the GEMM kernel
b) gpu_sim_insn = 154624 // 154624 instructions were executed
c) gpu_ipc = 28.8048 // the GPU achieve 28.8.48 IPC (instructions-per-cycle)
d) bw_util=0.002823 // DRAM Bandwidth utility is 0.002823
L2_total_cache_miss_rate = 0.3991// L2 cache hit rate is 0.6009
e) total FLOPs=(32*2-1)*32*32 = 64512
the simulation takes 5368/700M=7.66 us
64512/7.66 us =8.42*10^9 FLOPS/s

Question 6: Draw and discuss the following graphs:
a) Execution time (in seconds) that the kernel took
(hint: calculated as gpu_sim_cycles/core_frequency[Hz])
b) FLOP/s as a function of problem size (N)
c) IPC as a function of problem size (N)
d) Simulation time as a function of problem size (N)
e) External bandwidth as a function of problem size (N)
f) GPU occupancy as a function of problem size (N)

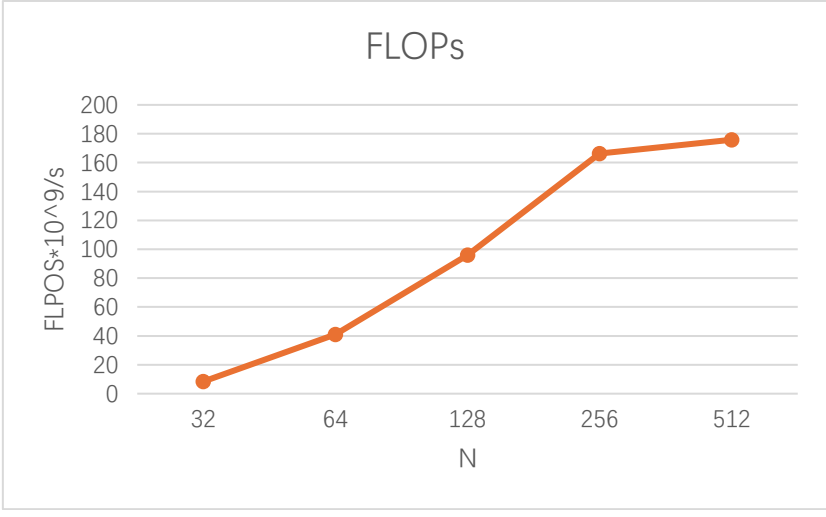
	Execution time/s(*10^(-6))	FLOP(*10^9)/s	IPC	Simulation time/s	External bandwidth	GPU occupancy
32	7.67	8.41	28.8048	1	0.002823	65.90%
64	12.7	40.96	121.6086	1	0.001363	66.10%
128	43.47	96.11	263.8152	8	0.0003983	66.35%
256	201.48	166.21	437.7152	65	0.0001289	66.49%
512	1525.05	175.85	453.3014	533	0.1424	66.58%

a)



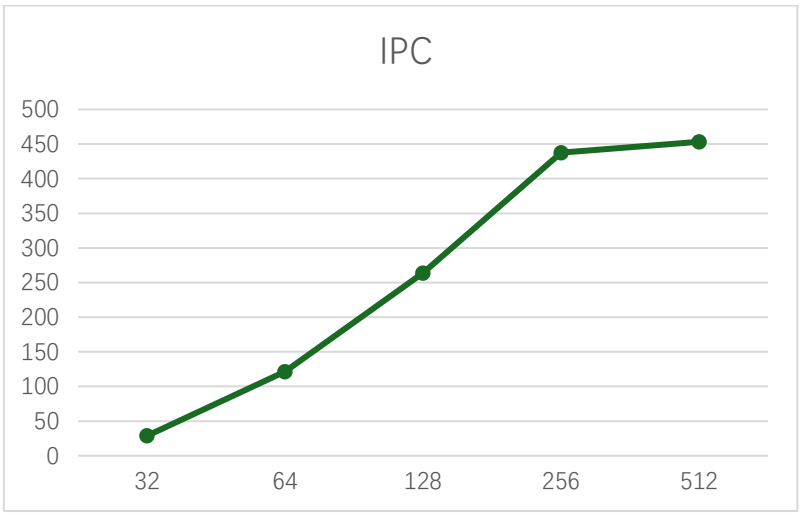
As N increases, the execution time also increases. This indicates that as the matrix size increases, more computational resources are required, leading to longer execution times.

b)



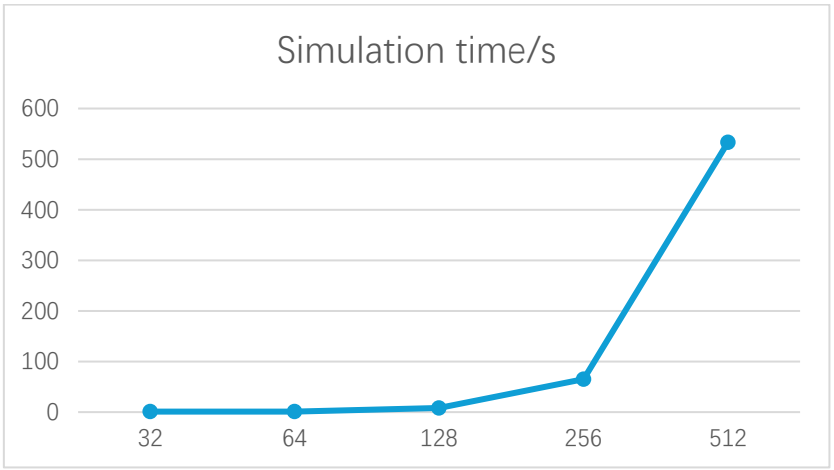
As N increases, the FLOP/s performance also increases. This indicates that the process benefits from larger matrix sizes in terms of computational efficiency.
There is a plateau in FLOP/s performance at larger values of N, particularly noticeable between N=256 and N=512. This indicates that beyond a certain point, increasing N does not significantly improve computational efficiency. This could be due to resource limitations, algorithmic constraints, or other factors.

c)



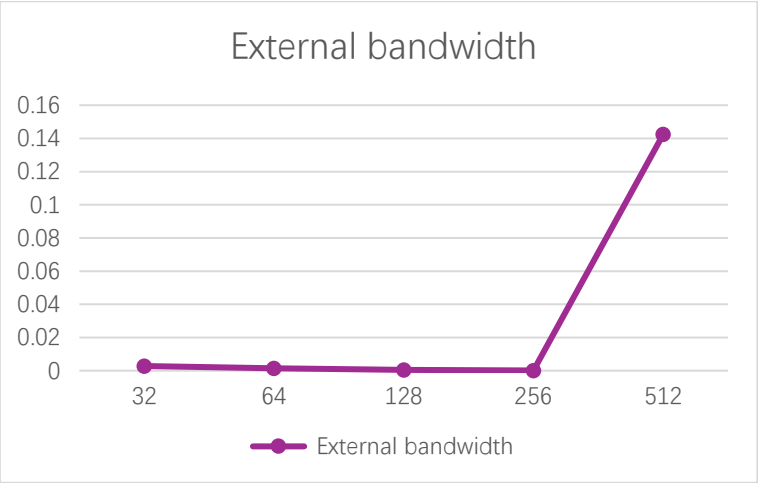
As N increases, the IPC(instruction per cycle)performance also increases. This indicates that the process benefits from larger matrix sizes in terms of computational efficiency.
IPC follows the same pattern as FLOP/s.

d)



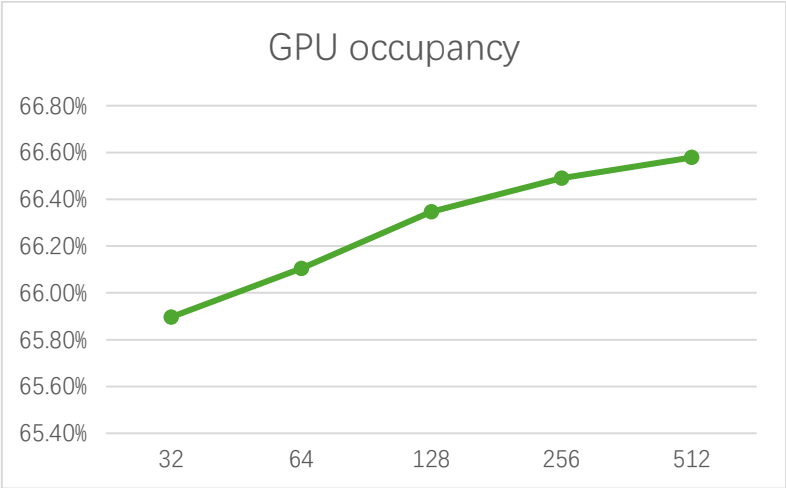
As N increases, the execution time also increases. This indicates that as the matrix size increases, more computational resources are required, leading to longer execution times. Since the simulation time is in proportion with the execution time, they follow the same pattern.

e)



As N increases, the external bandwidth decreases. This suggests that larger problem sizes require less data transfer between external memory and the processing unit per unit of time.
The external bandwidth value at N=512 is substantially higher compared to the previous values. This suggests a significant change in the algorithm or processing requirements at this point, potentially leading to a different memory access pattern or higher memory demands. It may come from that even though the miss rate drops when N=512, the data that draw from dram to cache become larger, then the bandwidth goes extremely higher.

f)



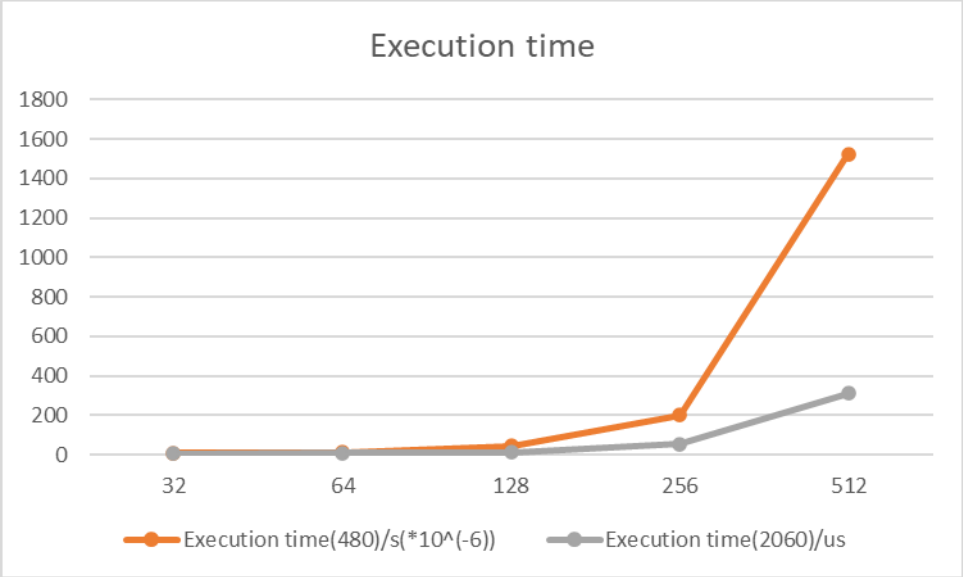
As N increases from 32 to 512, the GPU occupancy also increases gradually. This trend suggests that as the problem size or workload increases, the GPU is able to maintain a slightly higher percentage of active warps per cycle. The occupancy percentages range from 65.90% to 66.58%, indicating that the GPU is operating at a high level of occupancy. Occupancy values in this range suggest that the GPU is effectively utilizing its resources and is close to fully saturated.

Question 7: Repeat this exercise but with the NVIDIA RTX2060 GPU. Contrast your observation with the RTX2060 GPU against the prior observat RTX2060ions with the GTX480 GPU, and discuss and explain these differences and observations.
Frequency of NVIDIA RTX2060 is 1365 MHz

	Execution time/us	FLOP(*10^9)/s	IPC	Simulation time/s	External bandwidth	GPU occupancy
32	6.29	10.25	18.00	1	0	98.27%
64	8.39	61.94	94.34	2	0	99.02%
128	13.23	315.66	444.37	8	0	99.49%

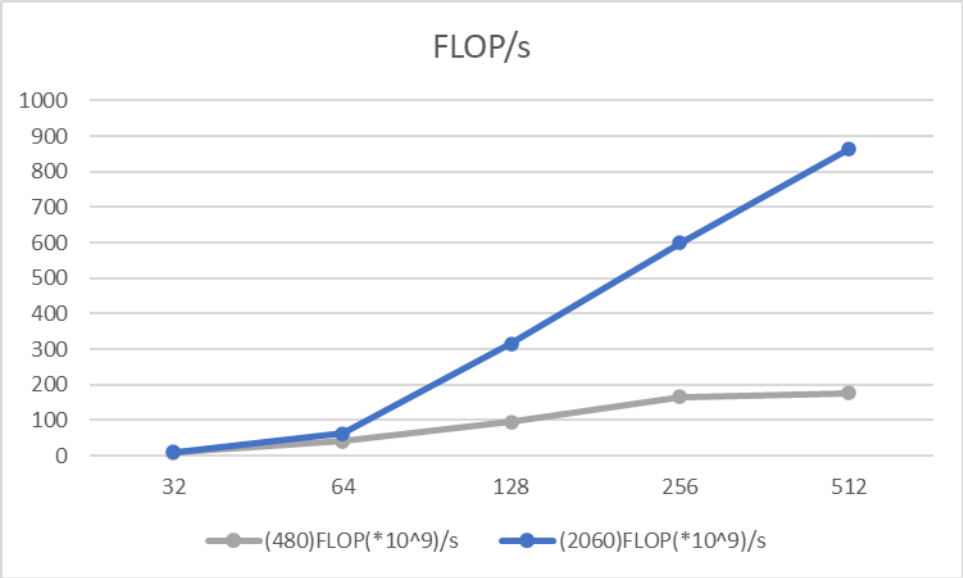
256	55.96	598.43	808.19	70	0	99.71%
512	310.65	863.24	1141.19	629	0	99.85%

a) Execution time (in seconds) that the kernel took



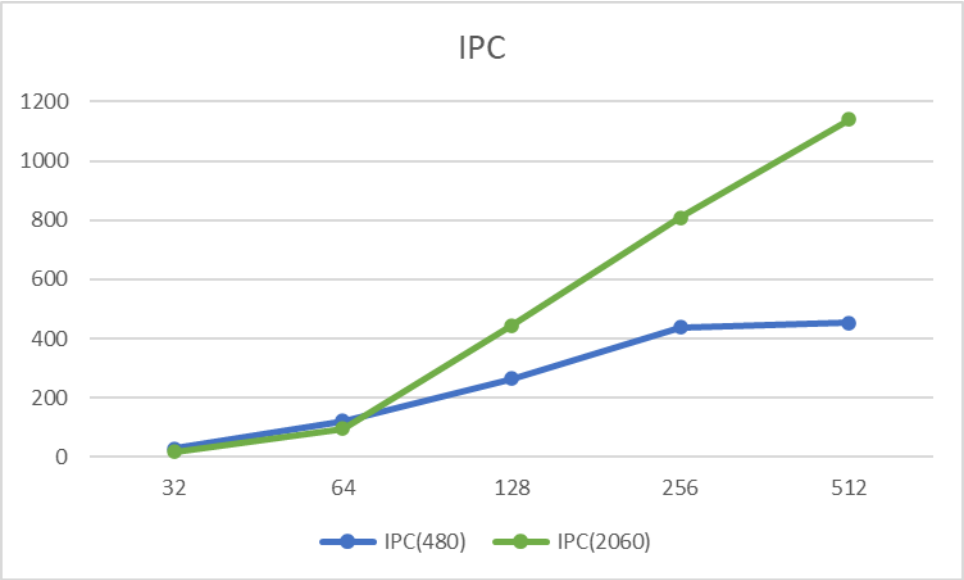
As N increases, the execution time also increases. This indicates that as the matrix size increases, more computational resources are required, leading to longer execution times. However, the execution time of RTX2060 is lower than GTX480. The RTX 2060 typically has more CUDA cores and operates at higher clock speeds compared to the GTX 480. This results in higher parallel processing capabilities and faster execution of compute-intensive tasks.

b) FLOP/s as a function of problem size (N)



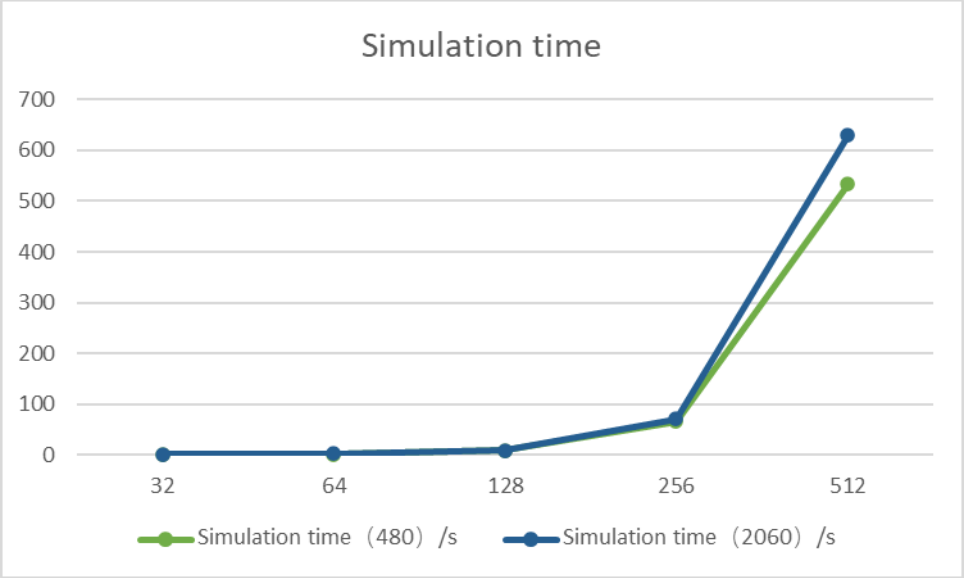
As N increases, the FLOP/s performance also increases. This indicates that the process benefits from larger matrix sizes in terms of computational efficiency.
There is a plateau over N=256 and this indicates that beyond a certain point, increasing N does not significantly improve computational efficiency for GTX480. However, RTX2060 hasn't reach its plateau until N=256, which means the best performance of RTX2060 is higher than that of GTX480.

c) IPC as a function of problem size (N)



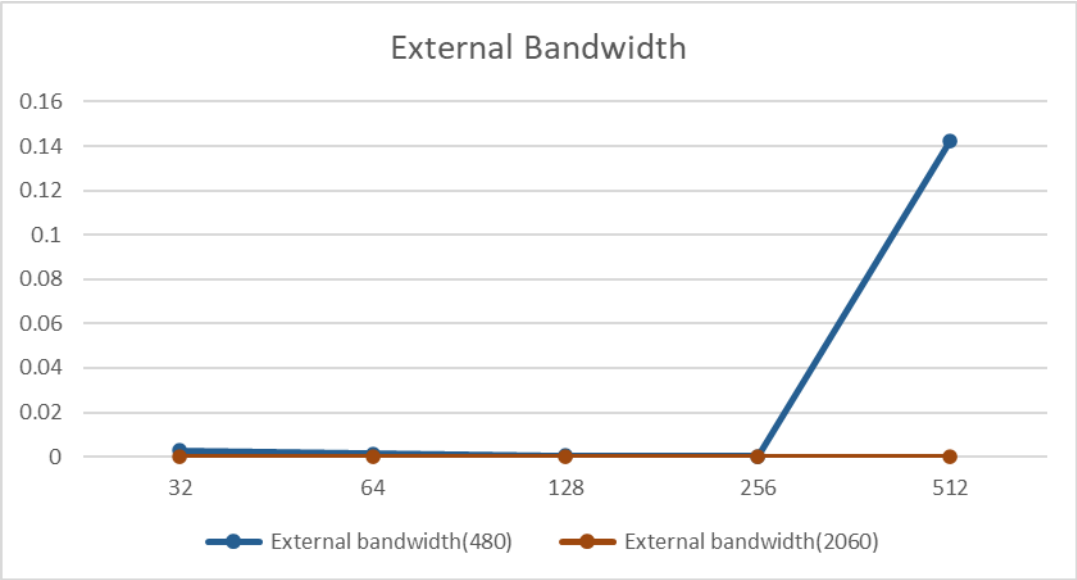
IPC follows the same pattern of FLOP/s. RTX2060 has better performance than GTX480 and the plateau is higher, which means at N=512, RTX2060 hasn't come to the saturation.

d) Simulation time as a function of problem size (N)



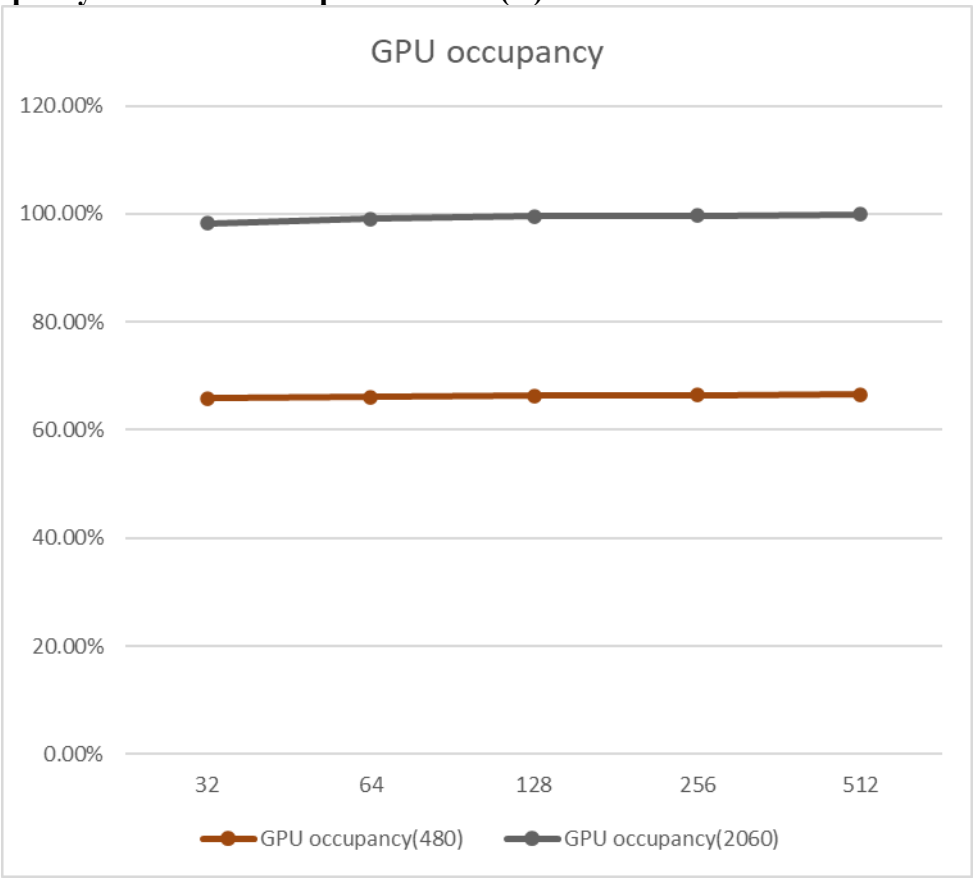
The simulation time is in proportion of execution time. As N increases, the execution time also increases. However, the proportion is not the same for GTX480 and RTX2060. Even though RTX2060 takes less execution time, it takes longer to simulate. Despite having faster memory, the RTX 2060 may face bottlenecks due to memory bandwidth limitations when handling large datasets or complex simulations. This can lead to longer data transfer times between the GPU and system memory, impacting overall simulation performance.

e) External bandwidth as a function of problem size (N)



The external bandwidth of GTX2060 is very small that it is almost zero. Because the hit rate is almost 100% so there's no need to access external memory.

f) GPU occupancy as a function of problem size (N)



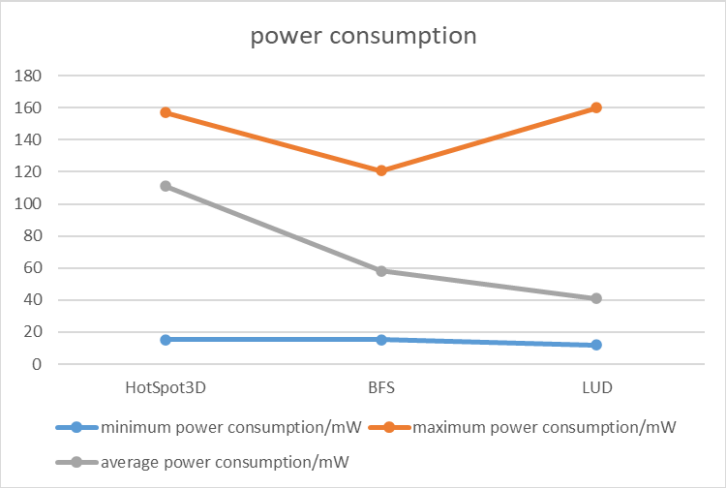
The GPU occupancy of RTX2060 is higher than that of GTX480 and it is close to 100%. The RTX 2060 typically has more CUDA cores and higher parallelism compared to the GTX 480. This increased parallelism allows the RTX 2060 to execute more concurrent threads, resulting in higher occupancy.

Question 8: Draw and discuss the following graphs. Use the GTX480 GPU model: a) Min, max, and average power consumption for the three applications, b) Execution time (in

seconds) that the three applications took (ignoring host overhead), Hint: Some applications might execute multiple kernels several times. Use the final gpu_tot_sim_cycle (instead of gpu_sim_cycle) metric for those. c) Energy consumption (Joules) of the three applications. Hint: Joules is the same as Watt-seconds (W * s).

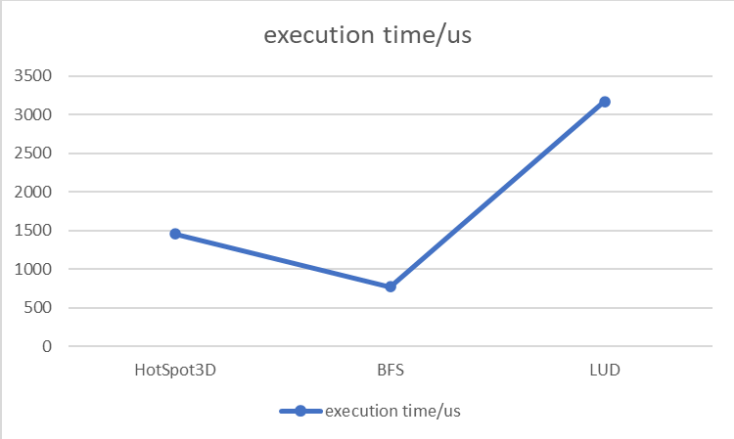
	minimum power consumption/mW	maximum power consumption/mW	average power consumption/mW	gpu_tot_sim_cycle	execution time/us	Energy consumption/nJ
HotSpot3D	15.25	156.937	110.978	1019148	1455.925714	161575.7239
BFS	15.25	120.788	58.0333	540882	772.6885714	44841.66767
LUD	12.066	160.068	40.8597	2219328	3170.468571	129544.3947

a) Min, max, and average power consumption for the three applications



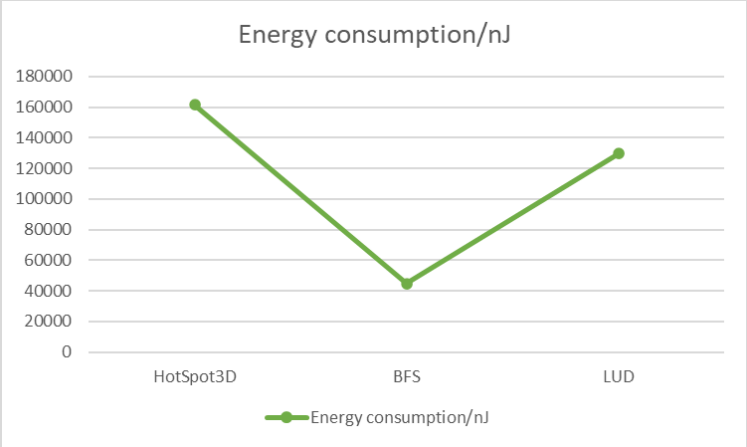
Across all workloads, the power consumption varies significantly, with HotSpot3D having the widest range (from 15.25 mW to 156.937 mW).On average, HotSpot3D consumes the highest power (110.978 mW), followed by BFS (58.0333 mW) and LUD (40.8597 mW).The minimum power consumption varies slightly among the workloads, with LUD having the lowest minimum power consumption (12.066 mW).HotSpot3D exhibits the highest maximum power consumption (156.937 mW), indicating that it may have periods of intense computational activity that require more power. Overall, the data suggests that different workloads have varying power consumption patterns, with some being more power-intensive than others.

b) Execution time (in seconds) that the three applications took (ignoring host overhead)



BFS has the shortest execution time, followed by HotSpot3D, and then LUD. This suggests that BFS is the least computationally intensive, while LUD is the most computationally intensive among the three workloads.

c) Energy consumption (Joules) of the three applications



BFS requires the least energy, while LUD requires the most energy among the three workloads.

Question 9: Use the GTX480 model and change the core frequency-, interconnect-,L2 clocks to 300 MHz and 1GHz respectively. Keep the DRAM clock unmodified. Redo question 8 but with the new clock frequencies. Discuss the result. Are the applications compute- or memory-bound?

300MHz

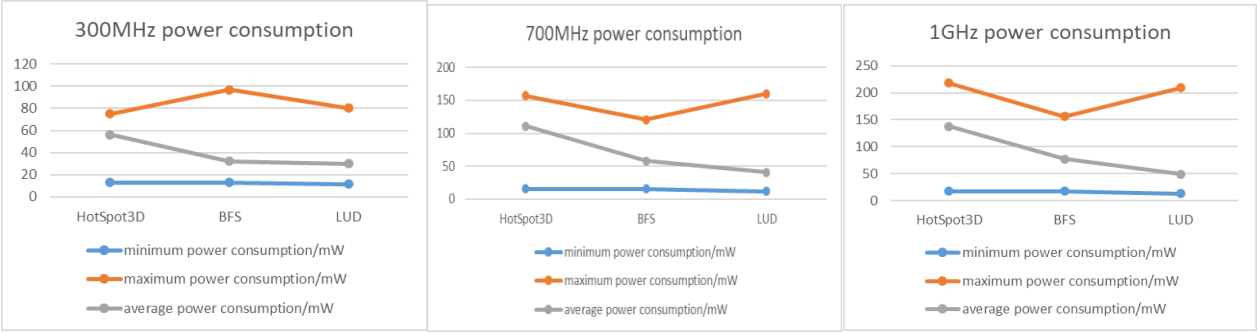
	minimum power consumption/mW	maximum power consumption/mW	average power consumption/mW	gpu_tot_sim_cycle	execution time/us	Energy consumption/nJ
HotSpot3D	12.8835	74.8936	56.2214	968874	3229.58	181571.509
BFS	12.8835	96.7819	32.0619	541414	1804.713333	57862.53842
LUD	11.6343	80.3007	30.118	2205317	7351.056667	221399.1247

1GHz

	minimum power consumption/mW	maximum power consumption/mW	average power consumption/mW	gpu_tot_sim_cycle	execution time/us	Energy consumption/nJ

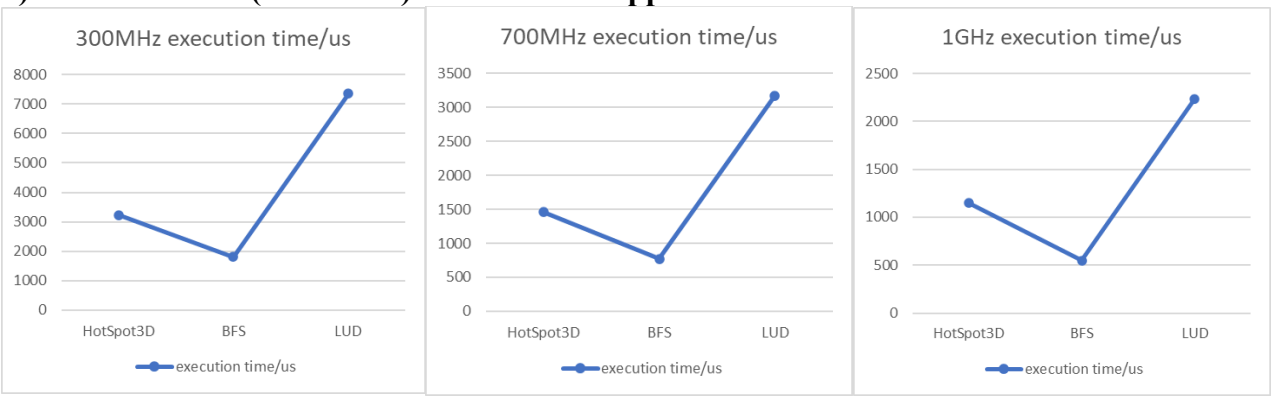
HotSpot3D	**17.0509**	**218.432**	**137.653**	**1147026**	**1147.026**	**157891.575**
BFS	**17.0509**	**156.485**	**77.0556**	**547715**	**547.715**	**42204.50795**
LUD	**13.1271**	**209.186**	**48.7689**	**2237942**	**2237.942**	**109141.9696**

a) Min, max, and average power consumption for the three applications



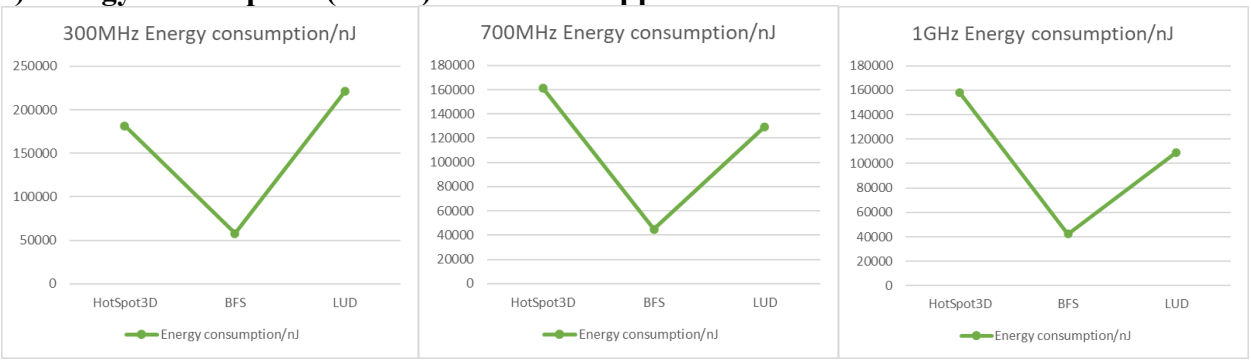
When the frequency goes higher, the power consumption gets bigger. However,they follow the similar pattern: HotSpot3D has the widest range and consumes the highest power, followed by BFS and LUD.The minimum power consumption varies slightly among the workloads, with LUD having the lowest minimum power consumption.HotSpot3D exhibits the highest maximum power consumption.

b) Execution time (in seconds) that the three applications took



When the frequency goes higher, the execution time gets shorter. With a higher frequency, the processor can execute more instructions per second, leading to faster completion of tasks.

c) Energy consumption (Joules) of the three applications



When the frequency goes higher, the energy consumption gets larger then gets lower. Because higher frequency means higher power consumption but lower execution time, which means at some point the energy consumption reaches a peak then goes down.

HotSpot3D and LUD are compute-bound, BFS is memory-bound. Compute-bound applications spend the majority of their time performing computations or executing instructions. As a result, they typically have longer execution times. On the other hand, memory-bound applications may exhibit shorter execution times if they are not heavily dependent on memory access.