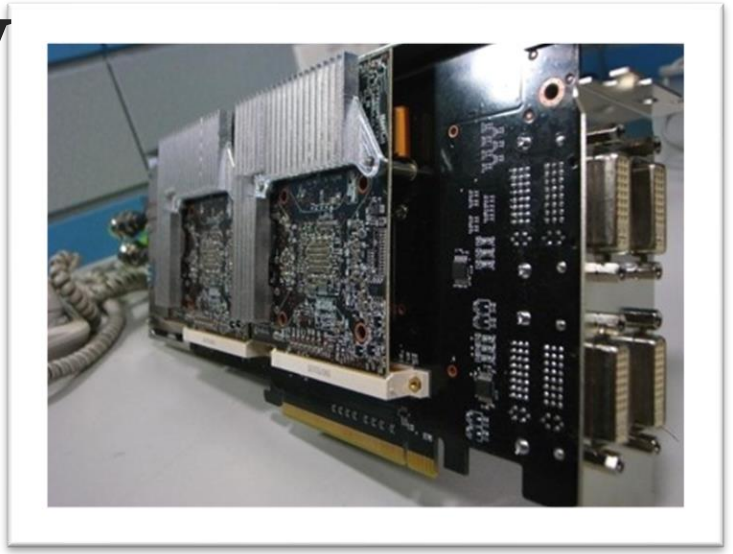
Multi-GPU Programming Pros and Cons: A Case Study

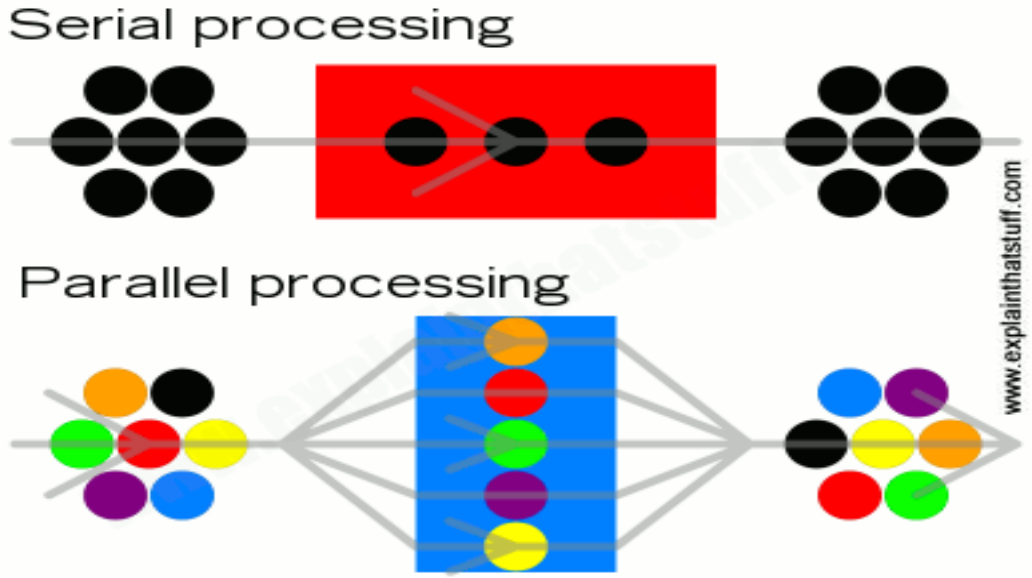


Ravishka Rathnasuriya
Dr. Eduardo Colmenares
Midwestern State University, Texas
Department of Computer Science

What is a Supercomputer?

- A computer with high level of performance compared to a general-purpose computer
- Use to perform parallel processing
- Knowledge of parallel computing is helpful to take advantages of the power of the supercomputers.

Accessed the Maverick2 Supercomputing Cluster at Texas Advanced Computing Center



What is a GPU?

- Graphics Processing Unit
- Part of modern Supercomputing
- Powerful in parallel computing. Ability to shrink large number and compute in seconds.
- Break complex problems into thousands or millions of separate tasks and work them out at once.
- Many cores. Number of operations processed per second is high. (high throughput).
- CUDA a parallel computing platform let programmers to take advantage of computing the power of GPUs.

Project Objective

- Write programs for Sequential , 1-GPU, 2-GPU versions.
- A study on the positive and negative aspects of using multiple GPU's in solving problems with larger data sets
- Identify the variations of program's performance with increasing size of data samples of Sequential vs 1-GPU vs 2-GPU programming.
- Achieving communication-communication and computation –computation parallelism between GPUs while moving from Multi-Core to Many-Core programming.
- Used C programming for sequential programming, and CUDA in C for GPU programming.

Device Properties

- Device name: GeForce GTX 1080 Ti
- Total global memory: 11721506816
- Size of shared Memory per block : 49152 bytes
- Number of registers per block: 65536
- Corresponding Warp Size: 32
- Maximum number of threads per block: 1024
- Maximum Number of threads that we can have for a 3D layout: x:1024 y:1024 z:64
- Maximum grid size: x:2147483647 y:65535 z:65535
- Number of Multiprocessors: 28

Software Used

- Visual Studio Code – Text Editor
- WINSCP
  Connect to the cluster
  Store files and execute jobs
- PUTTY
  Compile the programs
- TACC Visualization Portal
  To use NVIDIA Visual Profiler

Compilation and Execution Commands

1. Compilation Commands
   gcc <file_name.c> -o a.out
   nvcc <file_name.cu> -o a.out
2. Execution Commands
   sbatch <script_file>

Problem Sizes

Each Matrix has three contain three different sizes

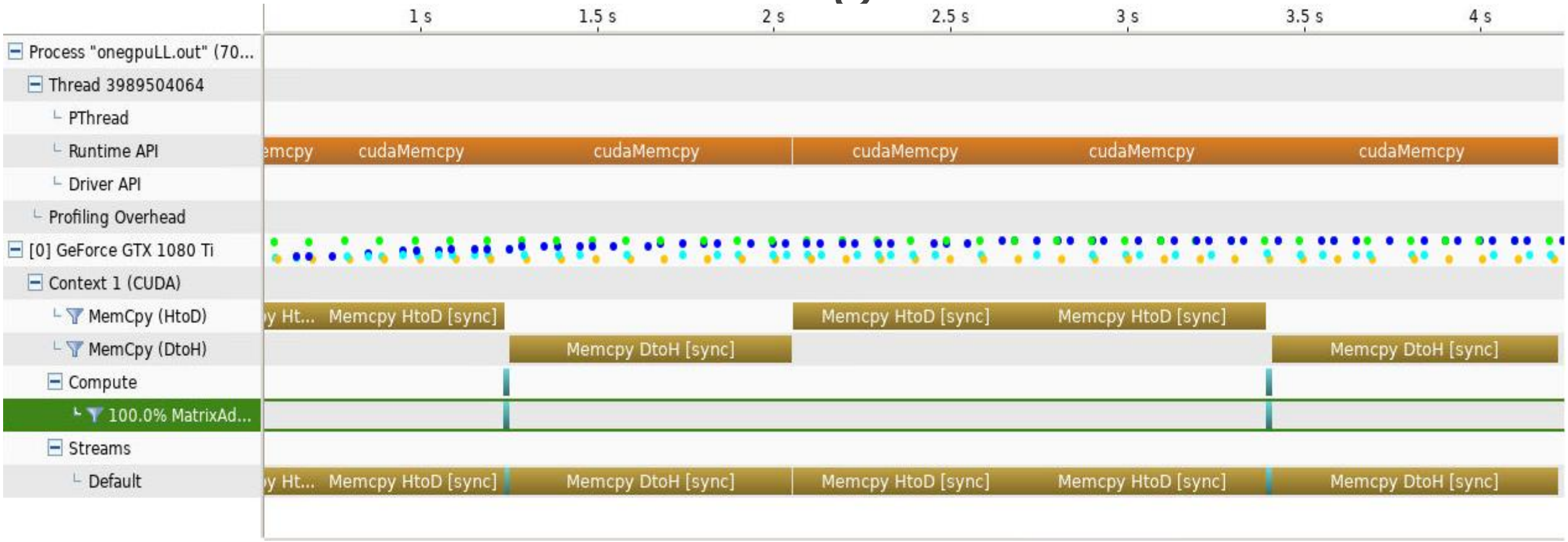
1. Small Size – 16384 elements
2. Medium Size – That fits the size of 1-GPU
   536,870,912 elements
3. Large Size – That goes beyond the size of 1-GPU
   1,073,741,824 elements

From 1-GPU to 2-GPU

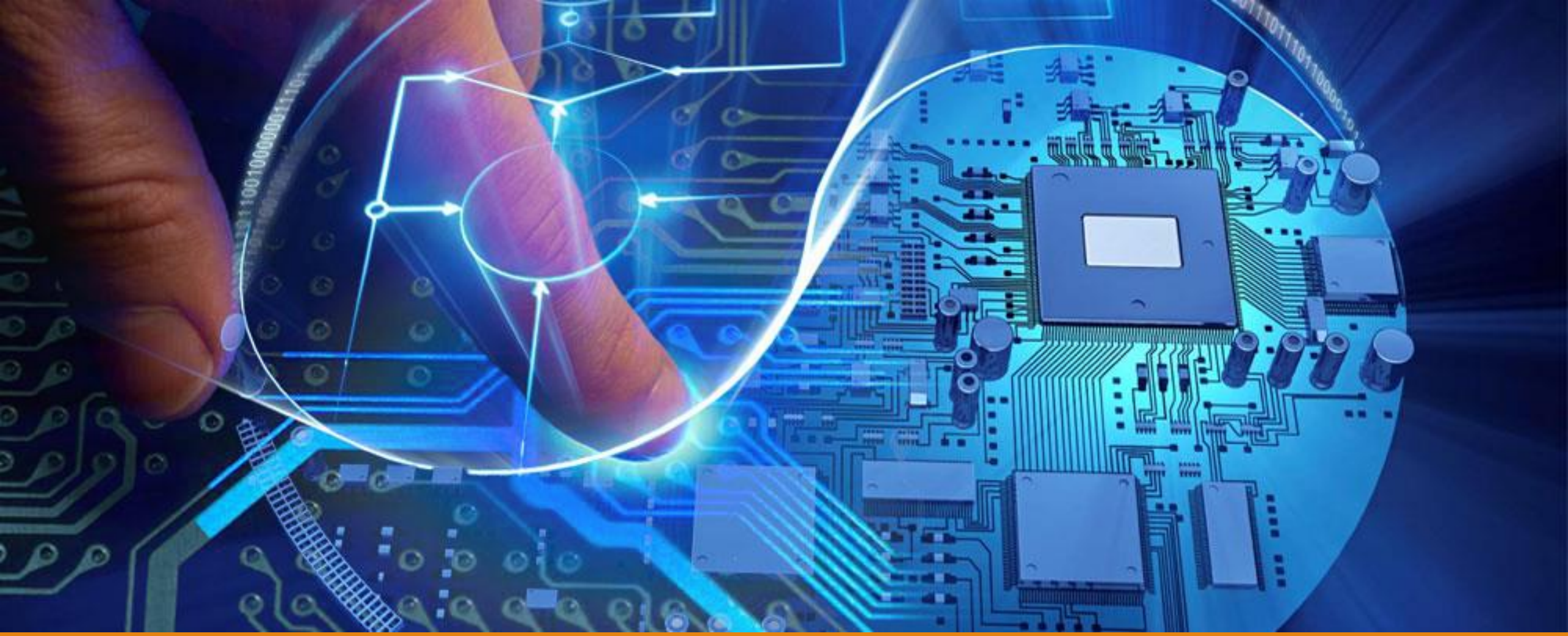
Conditions that should satisfy to obtain concurrency between multi-GPUs

1. Use non-default streams.
   Stream is a sequence of operations that execute on the device in order.
   ```
   cudaStream_t stream[2];
   cudaStreamCreateWithFlags(&stream[0],cudaStreamNonBlocking);
   MatrixAddition<<<dimGrid, dimBlock,0,stream[0]>>>(matAD[0],matBD[0], matCD[0]);
   ```
2. Use asynchronous commands for memory copies.
   ```
   cudaMemcpyAsync(matAD[0], &matAP[0*size/2], totalsize/2, cudaMemcpyHostToDevice,stream[0]);
   cudaMemcpyAsync(matAD[1], &matAP[1*size/2], totalsize/2, cudaMemcpyHostToDevice,stream[1]);
   ```
3. Host memory involved in data transfer must be pinned memory.
   ```
   cudaMallocHost((void**)&matAP,totalsize);
   ```

Visual Profiler for Large Size 1-GPU



Visual Profiler for Large Size 2-GPU

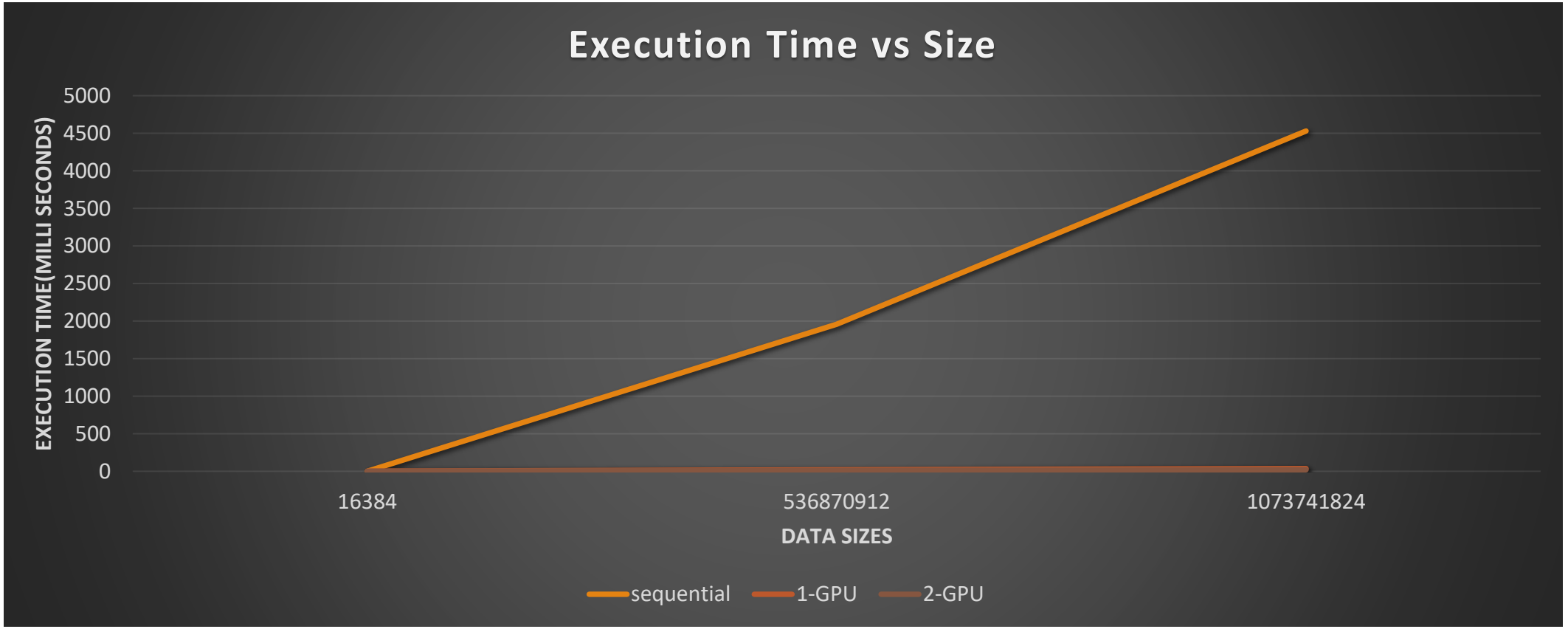


Performance Analysis

Execution Time(milli seconds)

Execution time (milli seconds)			
Method	**Problem Sizes (# of elements a matrix contain)**		
	16384	**536870912**	**1073741824**
Sequential	0.058463	1955.696896	4532.541298
1-GPU	0.030976	17.705025	35.487232
2-GPU	0.001632	8.84928	17.65814

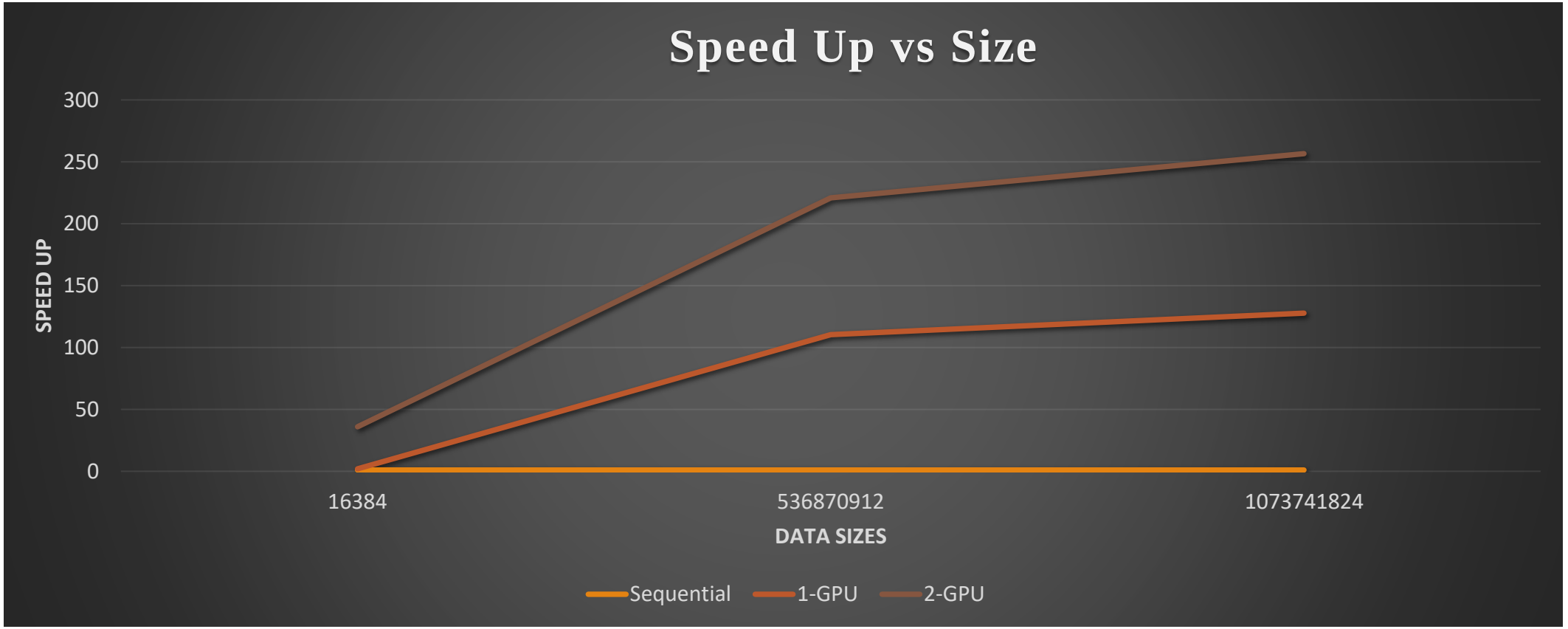
Execution Time Curve



Speed Up Table (Ts / Tp)

Speed up(Ts/Tp)			
Method	**Problem Sizes**		
	16384	**536870912**	**1073741824**
Sequential	1	1	1
1-GPU	1.887364411	110.4599907	127.7231568
2-GPU	35.82291667	221.0006798	256.6828272

Speed Up Curve



Pros of Multi-GPU Programming

- The execution time for a task becomes more faster than using 1-GPU in large data sets.
- 2 or more GPU'S can communicate with each other, reducing the time for communication between host-device
- Multi-GPUs can execute parallelly which optimized performance of the program
- Communication-communication and computation-computation parallelism can be achieved between GPUs.

Cons of Multi-GPU Programming

- CudaMallocHost that would allocate memory in the host would take more time.
- Power and energy for using Multi-GPUs is relatively higher than a single GPU

Acknowledgement

- Dr. Magaly Rincon
- Julie Scales and UGROW Committee
- Dr. Eduardo Colmenares
- Department of Computer Science, MSU
- Texas Advanced Computing Center
- NVIDIA