MATRIX ADDITION ON GPU

Submitted by : Goli Uma Sankar
Regd. No. : 18560

The overall structure of a CUDA program that uses the GPU for computation is as follows:

- First consider host (CPU) copies of a, b, c and their device (GPU) copies as d_a, d_b, d_c.
- Next, allocate memory in device and host. Allocate random data in vectors a and b (inside host). And then copy data to device.
- Launch **matadd()** kernel on device with N threads in N blocks (N is a fixed size of matrix).

  add_threads_blocks<<<(N + (THREADS_PER_BLOCK - 1)) / THREADS_PER_BLOCK, THREADS_PER_BLOCK>>>(d_a, d_b, d_c, N);
- In **matadd()** function, we run multiple threads in multiple blocks. While doing this seems unecessary, in some cases we need threads since they have communication (__shared__ variables) and synchronization (__syncthreads()) mechanisms.
- After the **matadd()** function completes, copy the computed values from the GPU device memory back to the host's memory.