Continuando con CUDA…

Multiplicación de Matrices Secuencial

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)  {
    c[i][j] = 0;
    for (k=0; k<N; k++)
      c[i][j] += a[i][k] * b[k][j];
  }
```

Otra forma…

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)  {
    for (k=0; k<N; k++)
      c[j+i*N] += a[k+j*N] * b[j+k*N];
  }
```

Multiplicación de Matrices Paralelo…

```
__global__ void matrixMultGPU(int *a, int *b, int *c) {
 int k, sum = 0;
 int col = threadIdx.x + blockDim.x * blockIdx.x;
 int fil = threadIdx.y + blockDim.y * blockIdx.y;

 if (col < N && fil < N) {
  for (k = 0; k < N; k++) {
   sum += a[fil * N + k] * b[k * N + col];
  }
  c[fil * N + col] = sum;
 }
}
```

```
__global__ void matrixMultGPU(int *a, int *b, int *c) {
int tx = threadIdx.x;
int ty = threadIdx.y;
float sum;
for (int k = 0; k < N; k++) {
        float Ael= a[ty * N + k];
        float Bel=  b[k * N + tx];
        sum += Ael*Bel;
  }
  c[ty * N + tx] = sum;
 }
}
```

- ¿Cuál es la diferencia entre las dos anteriores aproximaciones paralelas?
- ¿Como es el acomodo de hilos y bloques en cada uno de ellas?

Función Time - Speedup

Se puede utilizar para contabilizar el tiempo la función:

clock_t clock();

Speedup. Factor de mejora de rendimiento. También llamado aceleración o rapidez.

$$S(n) = T(1)/T(n).$$

$T(1)$→ Tiempo de ejecución del programa en secuencial

T(n) → Tiempo de ejecución del programa en paralelo

Eficiencia del Sistema

La eficiencia del sistema para un sistema con *n* procesadores se define como:

$$E(n)=S(n)/n = T(1)/n*T(n)$$

cudaDeviceSynchronization ()

Espera hasta que todos los comandos de procedimientos en todos los streams de los hilos del host han completado.

- Obtener el tiempo secuencial y paralelo de la multiplicación de matrices.
- Obtener el speedup
- Obtener la eficiencia del sistema