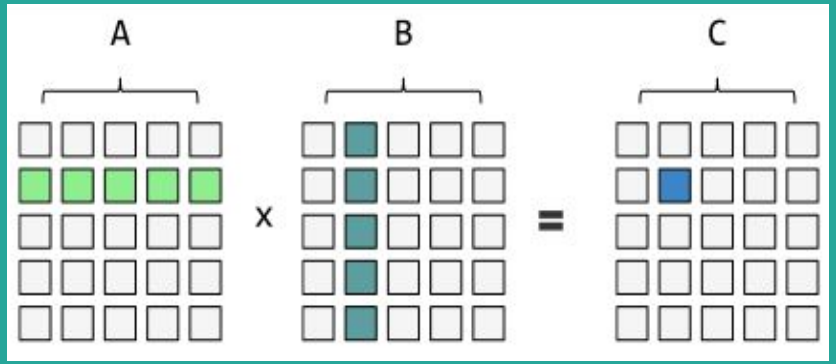
Programación paralela con el algoritmo de multiplicación de matrices

—

Por Sergio Rodríguez Rubio

¿Cómo optimizar el algoritmo de multiplicación de matrices usando programación paralela?

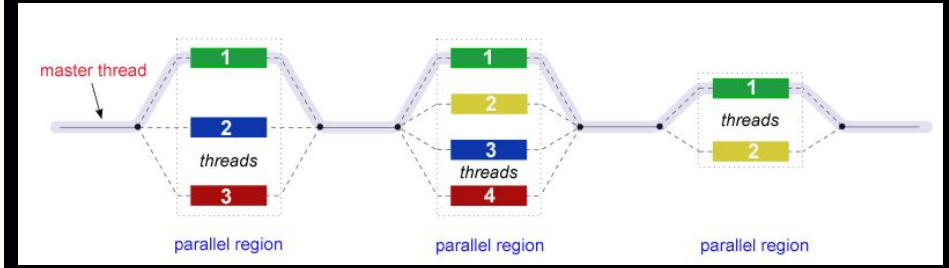



```
void Matriz_Matriz(float* x, float* y, float* z, int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            float num = 0.;
            for (int k = 0; k < size; k++) {
                num += x[i*size+k] * y [k*size+j];
            }
            z[i*size+j] = num;
        }
    }
}
```

Métodos de Paralelismo Estudiados

OpenMP®

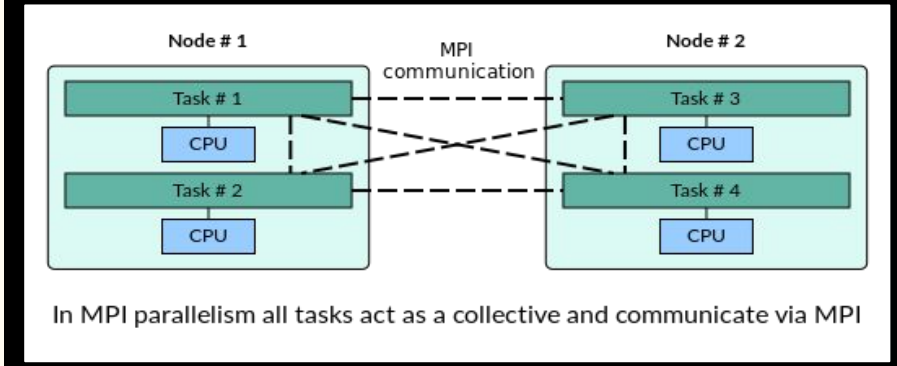
Open Multi-Processing




```
void Matriz_Matriz((float* x, float* y, float* z, int size) {
    int i, j, k;
    float num;
    #pragma omp parallel shared(x, y, z) private(i, j, k, num)
    {
        #pragma omp for schedule(static)
        for (i = 0; i < size; i++) {
            for (j = 0; j < size; j++) {
                num = 0.;
                for (k = 0; k < size; k++) {
                    num += x[i*size+k] * y [k*size+j];
                }
                z[i*size+j] = num;
            }
        }
    }
}
```

OPEN MPI

Open Message Passing Interface

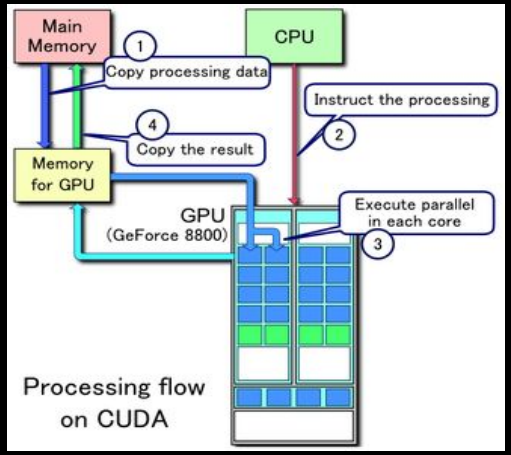



```
void Matriz_Matriz(float* x, float* y, float* z, int size, int rows, int offset) {
   for (int i = offset; i < offset+rows; i++) {
      if((i+1) > size) {
         break;
      }
      for (int j = 0; j < size; j++) {
         float num = 0.;
         for (int k = 0; k < size; k++) {
            num += x[i*size+k] * y [k*size+j];
         }
         z[i*size+j] = num;
      }
   }
}
```

NVIDIA

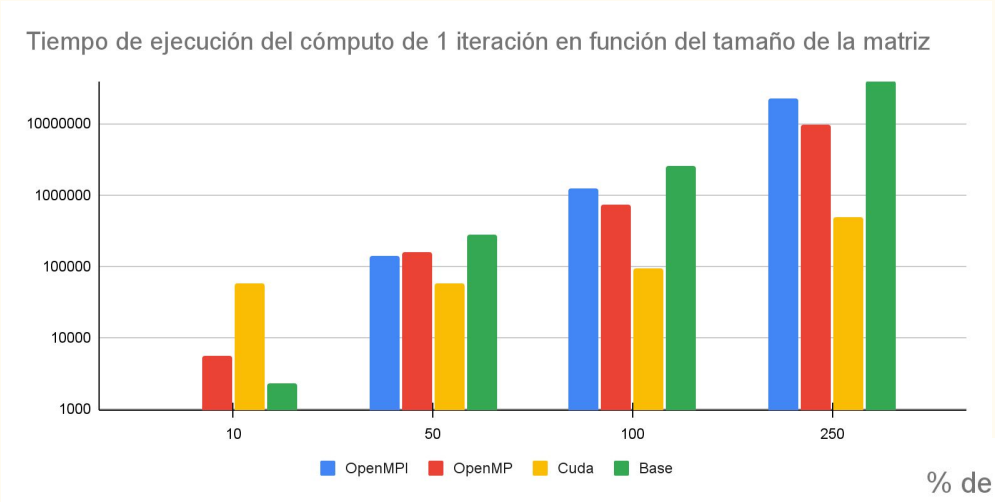
CUDA

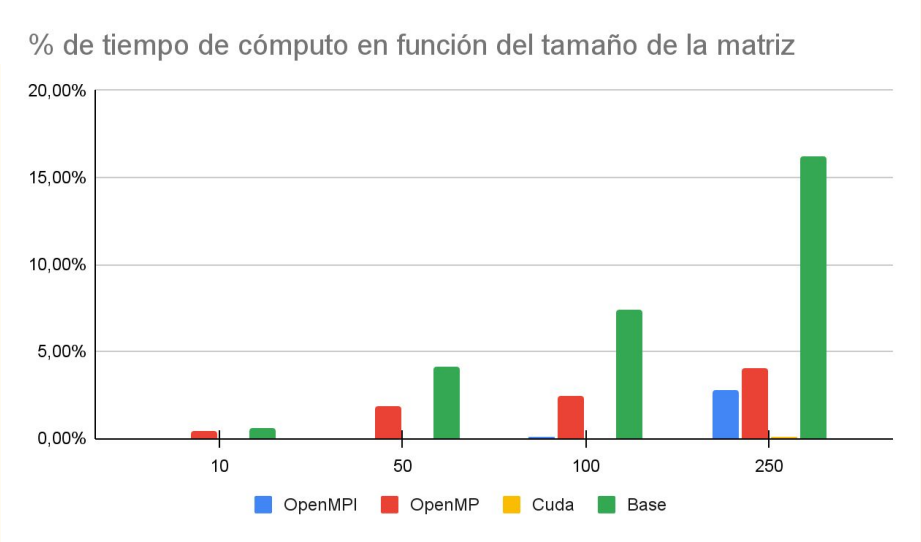
Compute Unified Device Architecture



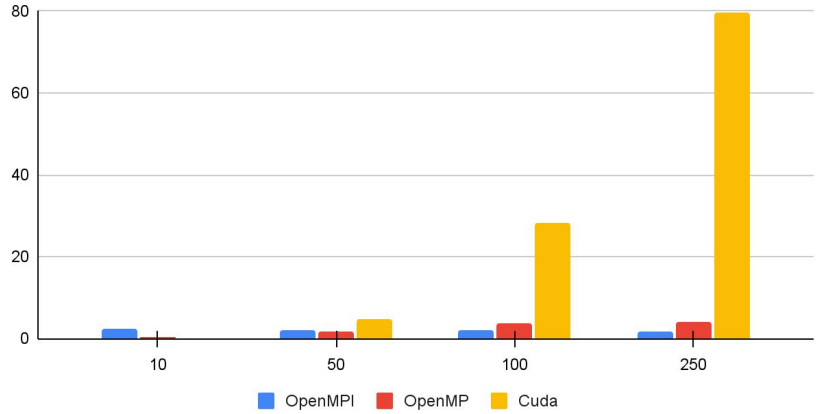

```
__global__ void Matriz_Matriz_kernel(float* x, float* y, float* z, int size) {
    int i = blockIdx.y * blockDim.y + threadIdx.y;
    int j = blockIdx.x * blockDim.x + threadIdx.x;
    float num = 0.;
    if (i < size && j < size) {
        for (int k = 0; k < size; k++) {
            num += x[i*size+k] * y [k*size+j];
        }
        z[i*size+j] = num;
    }
}
```

Resultados obtenidos

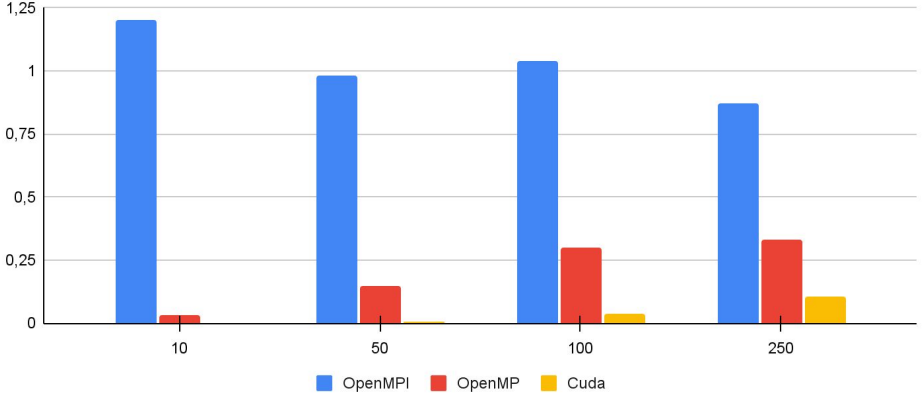

Tiempo de ejecución del cómputo de 1 iteración en función del tamaño de la matriz
10000000
1000000
100000
10000
1000
10
50
100
250
OpenMPI
OpenMP
Cuda
Base


% de tiempo de cómputo en función del tamaño de la matriz
20,00%
15,00%
10,00%
5,00%
0,00%
10
50
100
250
OpenMPI
OpenMP
Cuda
Base

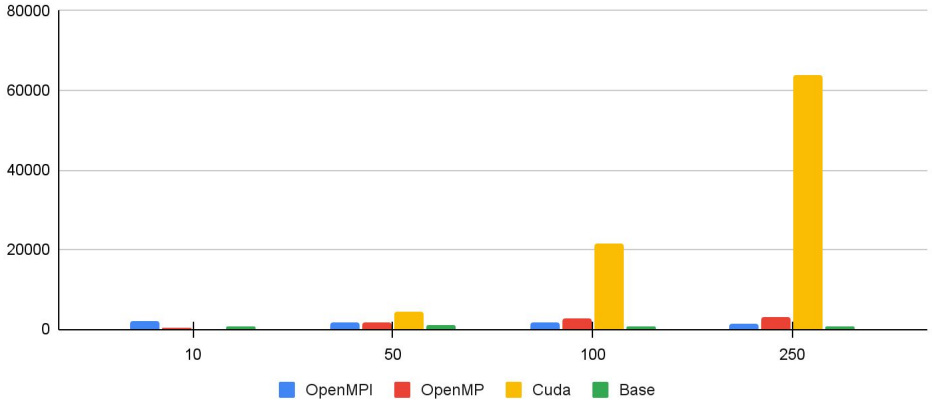
Speed up del cómputo en función del tamaño de la matriz



Eficiencia del paralelismo en el cómputo en función del tamaño de la matriz



MFLOPS en total en función del tamaño de la matriz



Unas conclusiones

¡Gracias por atender!

https://github.com/SergioRodriguezEnt/Paralelismo_En_MxM

GITHUB

MEMORIA