Informe Algoritmos Paralelos (multiplicación de matrices con y sin memoria compartida)

Luigy Machaca Arcana

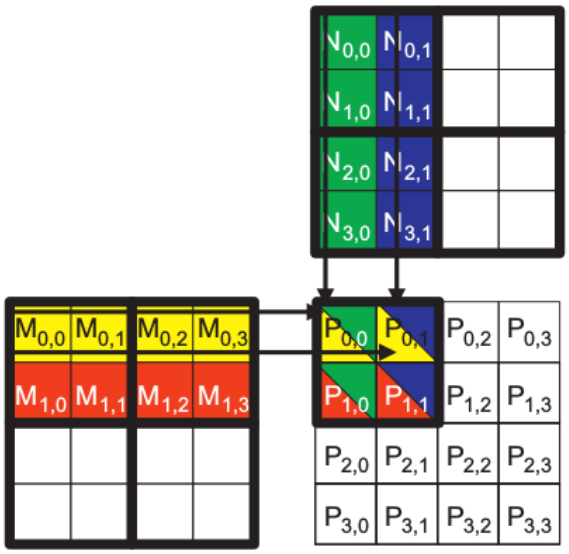
July 3, 2017

1 Tiled Matrix Multiplication

El proposito de utilizar Tile en la multiplicación de matrices o en cualquier otra operación que se quiera realizar en CUDA es reducir el tráfico de acceso a la memoria global, el cual es mas costoso que acceder a la memoria compartida.

La reducción del tráfico a la memoria global in la multiplicación de matrices es proporcional a la dimencion de bloques usados. Por ejemplo si usamos bloques de $16x16$, el tráfico a la memoria global puede ser potencialmente reducido a 1/16 a travez de la colaboración entre threads.

En la siguiente figura se muestra como es que la multiplicación de matrices con tail funciona. Se divide la matriz en tiles de $2x2$. El cálculo del producto punto realizado por cada thread ahora se dividira en fases. En cada fase, todos los threads en un bloque colaboran para cargar un tile de M y un tile de N en la memoria compartida.



2 Multiplicación de matrices sin tile

En esta sección se presenta el código de la multiplicación de matrices sin tile.

```
__global__
void matrix_mult(int** dd_mat_a,int** dd_mat_b,int** dd_mat_c, int n, int m){
        int value=0;
        int x = threadIdx.x + blockIdx.x*blockDim.x;
        int y = threadIdx.y + blockIdx.y*blockDim.y;
        if( y>n && x>m ) return;
```

```
        int i;
        for(i=0;i<m;i++){
                value += *(*(dd_mat_a)+y*m+i)   * *(*(dd_mat_b)+i*m+x);
        }

        *(*(dd_mat_c)+y*m+x)=value;


}
```

3 Multiplicación de matrices con tile

En esta sección se presenta el código de la multiplicación de matrices con tile.

```
__global__
void matrix_mult_shared(int** dd_mat_a,int** dd_mat_b,int** dd_mat_c, int width){

        __shared__ int Mds[WIDTH_TILE][WIDTH_TILE];
        __shared__ int Nds[WIDTH_TILE][WIDTH_TILE];

        int bx=blockIdx.x;
        int by=blockIdx.y;

        int tx=threadIdx.x;
        int ty=threadIdx.y;



        int value=0;

        int fil = by*WIDTH_TILE+ty;
        int col = bx*WIDTH_TILE+tx;

        int m;
        int k;
        for(k=0 ; k<width/WIDTH_TILE ; k++){
                Mds[ty][tx]=dd_mat_a[fil][k+WIDTH_TILE+tx];
                Nds[ty][tx]=dd_mat_b[k*WIDTH_TILE+ty][col];
                __syncthreads();

                for(m=0;m<WIDTH_TILE;m++){
                        value+=Mds[ty][m]*Nds[m][tx];
                }
                __syncthreads();

        }
        dd_mat_c[fil][col]=value;
}
```

4 Análisis

En esta sección se muestra los tiempos obtenidos ejecutando los códigos mencionados en las secciones anteriores.

En la siguiente se hace la comparacion de tres casos:

- sin memoria compartida

- con memoria compartida con una TILE de tamaño 35
- con memoria compartida con una TILE de tamaño 70