Algorítmos Paralelos
"Matrix Multiplication Kernel using shared memory"

Vanessa Santillana

22 Junio 2017

1 Análisis del los algoritmos

multiplicación de matriz de bloque que utiliza memoria compartida para reducir el tráfico a la memoria global.

```
1  __global__
2  void MatrixMulNewKernel(float* d_M, float* d_N, float* d_P,int
       Width) {
3    __shared__ float Mds[TILE_WIDTH][TILE_WIDTH];
4    __shared__ float Nds[TILE_WIDTH][TILE_WIDTH];
5    int bx = blockIdx.x; int by = blockIdx.y;
6    int tx = threadIdx.x; int ty = threadIdx.y;
7
8    int Row = by * TILE_WIDTH + ty;
9    int Col = bx * TILE_WIDTH + tx;
10   float Pvalue = 0;
11
12   for (int ph = 0; ph < Width/TILE_WIDTH; ++ph) {
13     Mds[ty][tx] = d_M[Row*Width + ph*TILE_WIDTH + tx];
14     Nds[ty][tx] = d_N[(ph*TILE_WIDTH + ty)*Width + Col];
15     __syncthreads();
16     for (int k = 0; k < TILE_WIDTH; ++k) {
17       Pvalue += Mds[ty][k] * Nds[k][tx];
18     }
19     __syncthreads();
20   }
21
22   d_P[Row*Width + Col] = Pvalue;
23 }
```

Este algoritmo mejora la operación de multiplicación de matrices, de la manera siguiente, primero declara matrices del tamaño del bloque, Mds y Nds de memoria compartida estática, después guardan los valores de threadIdx y blockIdx en variables automáticas en registros para un acceso rápido. Y el alcance es individual al hilo, y una vez que el hilo termina, los valores de estas variables dejan de existir. Luego se determinan los índices de fila y columna del elemento P que se produce por el hilo, después de itera con respecto a el tamaño de la matriz

sobre el tamaño de bloque, de tal manera que carga el elemento M apropiado en la memoria compartida. Puesto que ya conocemos la fila de M y la columna de N para ser procesada por el hilo.

La barrera __syncthreads() asegura que todos los hilos hayan terminado cargando el bloque M Y N en Mds y Nds antes de que cualquiera de ellos pueda moverse adelante.
El segundo bucle se encargara de realizar la operación de multiplicación en una variable, seguido de de otra barrera que asegurará que todos los hilos han terminado de usar los elementos M y N en la memoria compartida antes de que cualquiera de ellos se mueva a la siguiente iteración.

Finalmente se copia el valor resultado en P que tendrá la matriz resultante.

2 Tabla comparativa

TAMAÑOS		ALGORITMOS	
MATRIX	BLOCK	MULMATRIX basic	MULMATRIX
100	10	0.371743	0.353465
	100	0.363175	0.347963
	500	0.367996	0.337178
5000	10	1.371606	1.098069
	100	1.054113	1.115904
	500	1.107083	1.092676
10000	10	11.914652	7.853413
	100	12.044038	7.815705
	500	11.872794	7.83741

Esta tabla demuestra que existe una gran diferencia al resolver una matriz de tamaño 10000 con el algoritmo que usa bloques, con una mejora en tiempo mucho mayor que la multiplicación básica, sin embargo con un tamaño mas menor los algoritmos tienen casi tiempos iguales, pero aun así la multiplicación por bloques presenta una mejora.