Tiled Matrix Multiplication

Christofer Fabián Chávez Carazas
Universidad Nacional de San Agustín
Algoritmos Paralelos

25 de junio de 2017

1. Problema

La *Tiled Matrix Multiplication* optimiza el acceso a memoria global de la multiplicación de matrices convencional. Esta técnica consiste en dividir la matriz en pequeños bloques llamados *tiles*. En cada *tile*, los hilos cooperan entre sí para cargar los elementos del *tile* de memoria global a memoria compartida, para luego utilizar esos elementos en las operaciones que cada hilo hace.
Para poder optimizar al cien por ciento la multiplicación de matrices, tenemos que resolver dos cuestiones: El tamaño de la grilla y el tamaño óptimo de los *tiles*. Sólo los hilos de un mismo bloque pueden acceder a la misma memoria compartida, en otras palabras, cada bloque tiene su memoria compartida. Como los hilos van a guardar los valores de una *tile* dentro de la memoria compartida, entonces, para que no haya ningún error en las operaciones, el tamaño de la grilla en la primera dimensión es $n/TILE\_WIDTH\ x\ n/TILE\_WIDTH$ y en la segunda dimensión es $TILE\_WIDTH\ x\ TILE\_WIDTH$; siendo n el tamaño de la matriz (matriz cuadrada) y $TILE\_WIDTH$ el tamaño de los *tiles*.
Para poder conseguir el tamaño óptimo de los *tiles* se necesita saber las propiedades de la tarjeta CUDA, más específico, se necesita saber el tamaño de la memoria compartida. Debido a que todo un *tile* es guardado en memoria compartida, entonces el tamaño del *tile* no debe exceder al tamaño de la memoria compartida, ni tampoco ser tan pequeño. Para obtener las propiedades del dispositivo se utiliza la función $cudaGetDeviceProperties$ a la cual se le pasa por referencia una estructura $cudaDeviceProp$ en donde se guardarán las propiedades. Luego se obtiene la variable $sharedMemPerBlock$ de esa estructura.
Además de saber el tamaño de la memoria compartida, también se necesita saber cuántos hilos caben en cada bloque, para no entorpecer las operaciones en lugar de optimizarlas. Este máximo se consigue con la variable $maxThreadsPerBlock$ de la estructura $cudaDeviceProp$.

2. Experimento

Para los experimentos se está usando la GPU de La Salle con las siguientes propiedades:

	Tamaño de la matriz			
	1024	**2048**	**4096**	**8192**
Normal	5.187	43.968	358.529	2729.803
Tiled	1.906	13.471	105.73	858.096

Cuadro 1: Tiempos de los algoritmos para la multiplicación de matrices en milisegundos

- $sharedMemPerBlock$: 49152
- $maxThreadsPerBlock$: 1024

Se eligió 32 como tamaño de los *tiles*, ya que cumple con no sobrepasar el máximo de hilos por bloque ($32*32 = 1024$) y cumple con no sobrepasar el tamaño de la memoria compartida ($32 * 32 * 4 = 4096$). Los tiempos se muestran en la Tabla 1. Los tiempos están dados en milisegundos. Se puede observar una gran diferencia de tiempo entre el algoritmo *tiled* contra el algoritmo convencional.