Singular Value Decomposition on GPU using CUDA

Sheetal Lahabar & P.J. Narayanan

El artículo presenta la implementación de la descomposición de valores singulares (SVD, por sus siglas en inglés) para matrices densas usando un CUDA para programar una GPU. En particular, compara el desempeño del algoritmo frente a otros ya implementados en CPU como son uno de MATLAB y otro de Intel, consiguiendo superar a ambos algoritmos en cuestión de desempeño.

Existen dos razones importantes por las que los autores decidieron generar esta implementación; la primera es que el cálculo de SVD es una forma de tener un algoritmo robusto respecto a los errores numéricos al factorizar una matriz, y la segunda se debe que existen un gran número de aplicaciones en diversas áreas de estudio.

El argumento principal para usar una GPU que realice los cálculos del SDV en lugar de la CPU es el rápido incremento en desempeño que se han dado en las tarjetas gráficas, los avances en la generación de diversas librarías de algebra para CUDA (CUDABLAS y LAPACK), los diversos trabajos realizados por diversos investigadores que han encaminado la implementación del SDV en GPU, así como las diversas limitaciones que existen en cuestión a la memoria y poder de procesamiento en una CPU.

En términos generales el algoritmo se divide en 3 pasos. Dada la matriz $A \in M_{mxn}$ se debe de:

1. Descomponer la matriz en un producto de una matriz bidiagonal B con dos matrices unitarias tales que $A = QBP^T$, donde la matriz Q es de mxm, la matriz P es de nxn y la matriz B es de mxn.
2. Diagonalizar la matriz B, es decir, encontrar matrices X de mxm y Y de nxn tales que $\Sigma = X^T BY$.
3. Calcular los valores singulares de la matriz.

Cabe destacar que el proceso anterior implica subdividir la matriz en bloques de diversos tamaños, mandar cierta información a la CPU para procesos secuenciales y un buen aprovechamiento de los recursos de computo. Como, por ejemplo, que las operaciones de multiplicación de matrices y vectores en CUBLAS es óptima si las dimensiones son múltiplos de 32, esto debido a la estructura física de la memoria, por lo que agregan “ceros” a la matriz hasta obtener una cuya dimensión cumpla dicho criterio. También el identificar cuándo un proceso es únicamente secuencial y en ese momento aprovechar el procesamiento de la CPU para realizar dicha etapa.

En particular, me llama mucho la atención como generaron un algoritmo hibrido al realizar la primera etapa en la GPU ya que todo el proceso era paralelizadle, mientras que en la segunda etapa parte del proceso fue mandado a la CPU debido a que existía una parte que era forzosamente secuencial y aprovecharon el procesador para realizar esta tarea. Otra cuestión interesante es la forma en el manejo de la memoria compartida de la GPU, ya que para algunos tamaños de matrices era imposible cargar toda la información a esta y por ende tuvieron que optimizar el uso de la memoria para no tener problemas, esto habla de tener un buen entendimiento de la estructura física de la GPU que permita prever este tipo de complicaciones.

Víctor Augusto Samayoa Donado

CVU: 175750