# Arturo Gonzalez Bencomo 172906

## Resumen de paper: “Singular Value Decomposition on GPU using CUDA”

## Teoría

Este paper por Sheetal Lahabar y P. J. Narayanan expone un algoritmo para la descomposición SVD de una matriz utilizando cómputo paralelo mediante procesadores gráficos (GPU) y el software CUDA.

Comienza referenciando la importancia de la descomposición SVD y sus casos de aplicación: computacion matematica, ordenamiento, computación geométrica, multiplicación matricial, Fast fourier transform, etc.

Comienza referenciando el contexto histórico y evolución de los métodos previos tanto matemáticos como computacionales para llevar a cabo la descomposición SVD.

Tecnológicamente el desarrollo del cómputo masivo con procesadores gráficos, Field Programmable Gate Arrays (FPGAs) y Procesadores Cell ha tenido una gran impacto en la manera en que son desarrollados los algoritmos matemáticos. La taxonomía de Flynn es la arquitectura de referencia para el diseño de algoritmos en paralelo.

El paper hace énfasis en que el desarrollo en este tipo de tecnologías de cómputo paralelo va más rápido que lo predicho por Gordon Moore en su ley que dice: “aproximadamente cada dos años se duplica el número de transistores en un microprocesador”.

Posteriormente se explica el algoritmo Golub-Reinsch implementado. Escogieron este algoritmo dado que se puede mapear fácilmente a una arquitectura SIMD (Taxonomía de Flynn) de GPU.

Implementación de algoritmo secuencial:

Dada una matriz A, se realizan los siguientes pasos para encontrar su descomposición SVD:

$$1 : B \leftarrow Q^T AP \quad Bidiagonalización de A a B$$

$$2 : \Sigma \leftarrow X^T Y \quad Diagonalización de B a \Sigma$$

$$3 : U \leftarrow QX$$

$$4 : V^T \leftarrow (PY)^T Compute orthogonal matrices U and V^T and SVD of A = U \Sigma V^T$$

## Desarrollo

Este cómputo se puede realizar utilizando las operaciones de nivel 2 de BLAS, sin embargo computacionalmente es muy costoso ya que implica muchas lecturas y escrituras a memoria.

La implementación en paralelo los mismos pasos del algoritmo secuencial pero con CUBLAS. CUDA-BLAS (CUBLAS) proporciona operaciones de alto desempeño de multiplicaciones del tipo matriz-vector, matriz-matriz y funciones de cómputo de normas utilizando CUDA.

Hay factores importantes a considerar al desarrollar aplicaciones en CUDA:

Un factor importante a considerar es que CUBLAS tiene un mejor desempeño cuando opera con matrices de dimensiones múltiplos de 32 por aspectos de memoria. Ante esto se rellenan los vectores y matrices con ceros ajustando sus dimensiones a múltiplos de 32.

El desempeño de CUBLAS depende también de la transferencia de los datos entre el CPU y el GPU. Dado que en una computadora la memoria principal es la memoria RAM y el elemento de principal de procesamiento es el CPU, estos tienen control de la ejecución de los procesos y son estos quienes asignan los datos a la memoria del GPU, es muy relevante tomar en cuenta las velocidades de transferencia entre ellos. A modo de ejemplo la tarjeta NVIDIA 8800 GTX tiene tasas de transferencia internas de 86.4 GB/s mientras que la tasa de transferencia entre CPU y GPU es una orden de magnitud más lenta por lo que se tienen que minimizar la transferencia de datos entre CPU y GPU para maximizar el desempeño del algoritmo.

Para probar el algoritmo implementado se construyeron varios datasets con distintas características que a continuación mencionan:

1. Se generaron 10 matrices cuadradas aleatorias densas, cada algoritmo fue ejecutado 10 veces por matriz y se promedio el tiempo de cómputo por cada una de las matrices.
2. Prácticamente lo mismo pero enves de ser cuadradas se generaron matrices rectangulares, se ejecutó 10 veces el modelo y se promedio el tiempo de ejecución.

Se ejecutaron los algoritmos en tres plataformas una con CPU, y dos con GPUs y se realizaron mediciones en las distintas etapas del algoritmo obteniendo los resultados matemáticos y midiendo el tiempo de ejecución de los mismos.

Procesadores con los que se ejecutó el algoritmo.

| **Nombre** | **Capacidad Máxima** |
|---|---|
| GTX280 | 933 GFLOPS |
| Intel Core 2 Duo CPU E6750 | 22.4 GFLOPS |
| 8800 GTX: | 345.6 GFLOPS7 |

# Resultados:

Es fundamental destacar que las descomposiciones en SVD en CPU siguen siendo más rápidas que en GPU cuando se trata de matrices pequeñas pero a medida que se incrementa el tamaño de la matriz el desempeño es cada vez más rápido con el uso del GPU. Además se realizaron combinaciones de procesamiento tanto en CPU como en GPU en las distintas etapas para lograr rendimiento óptimo en las etapas que no son fácilmente paralelizables.

El algoritmo implementado sobre CUDA tuvo un desempeño de 3 a 8 veces más rápido que su equivalente operación en Intel MKL y entre 3 y 59 veces que MATLAB que se ejecuta de manera secuencial.

La diagonalización se llevó a cabo enteramente en el GPU.

La diagonalización se implementó combinando CPU y GPU.

El cómputo final se lleva a cabo con GPU.