Reporte de lectura: Oksa y Vajtersic (2007) Parallel One-Sided Block Jacobi, SVD Algorithm: I. Analysis and Design

Fernando Felipe Briseño Martínez

31 de mayo de 2018

Resumen

El artículo hace uso del método Jacobi de un lado para hacer el cálculo de la descomposición de valores singulares (SVD, por sus siglas en inglés) describiendo e implementando estrategia para acelerarlo haciendo uso de cómputo en paralelo.

El método de Jacobi es importante porque permite computar los valores singulares, así como los vectores singulares con alta precisión relativa. Esto cobra relevancia para disciplinas en las cuales valores pequeños deben ser calculados muy precisamente. La aportación de Oksa y Vajtersic consiste en hallar una implementación del método de Jacobi con la velocidad suficiente como para ser útil para estos fines.

Estructura del artículo

El documento se divide en cinco secciones:

1. Introducción
2. El algoritmo de Jacobi en bloque de un lado
3. Acelerando el algoritmo
4. Estrategia de paralelización
5. Conclusiones

La segunda sección hace una descripción del algoritmo de Jacobi, mientras que el tercero describe diversas estrategias utilizadas para reducir el número de operaciones de punto flotante requeridas para el algoritmo, mientras que el cuarto habla de su implementación usando Message Passing Interface (MPI).

Argumentación

El algoritmo de Jacobi en bloque de un lado (OSBJ, por sus siglas en inglés) se usa para el cálculo de la descomposición SVD de una matriz A de orden $m \times n$ donde $m \geq n$.

El OSBJA se puede escrbe como un proceso iterativo en el que se multiplica la matriz A para mutuamente ortogonalizar las columnas entre dos bloques-columna de A. Este proceso termina cuando la matriz A tiene columnas altamente ortogonales.

Estas actividades toman para cada paso $m(n_i n_j + n_i + n_j) + 8(n_i + n_j)^3 + 2m(n_i + n_j)^2$ operaciones de punto flotante.

Cada paso de este algoritmo consiste en tres partes:

1. Para cada valor $A_{i,j}$ y par i, j de vectores columna de A, se calcula la matriz simétrica, positiva semidefinida de producto punto entre cada par. Esta parte requiere de varios productos punto de tamaño m.

2. $A_{i,j}$ se diagonaliza, es decir se calcula la descomposición de eigenvalores. Esta puede ser rápida si se eligen bloques lo suficientemnte pequeños para hacer las operaciones en bloque.
3. Se actualizan los bloques columna de A. Esta es la parte más costosa en términos de cálculo, dado que requiere de varias operaciones matriciales.

El artículo señala que en la práctica los métodos Jacobi siempre convergen.

En la sección 3, los autores discuten diversas estrategias para acelerar el algoritmo. En específico, señala dos grandes áreas de mejora: el preprocesamiento de la matriz A y el uso de lo que denominan transformaciones ortogonales en bloque rápidamente escaladas.

En cuanto a la primera estrategia, los autores proponen hacer una factorización QR de A seguida de una factorización LQ del factor R resultante del paso previo. Esto provoca una reducción considerable de los pasos en el algoritmo Jacobi, especialmente las actualizaciones ortogonales matriciales. Asimismo, los autores proponen incializar ciertas matrices usadas durante el proceso Jacobi de tal forma que las columnas dentro e cada bloque de columnas sea mutuamente ortogonal, de tal forma que se puedan aprovechar para la segunda estrategia.

La seguna estrategia busca romper el cálculo para usar matrices pequeñas para todas las actualizaciones par apoder usas la memoria caché.

De esta forma, el algortimo se puede acelerar de tal forma que cada paso puede ser varias veces más rápido que usando el camino estándar, llevando la mayor parte del cálculo a la memoria caché.

Finalmente, los autores describen una estrategia para la paralelización de este método. Ellos deciden usar MPI y las librerias BLCS para comunicación, y la librería ScaLAPACK para el cómputo distribuido. LAPACK es usado para el cómputo dentro de cada procesador.

Los autores señalan que las tareas se pueden dividir fácilmente entre procesadores sin necesidad de comunicación hasta el momento en el que se implementa un criterio para detener el algoritmo.

Perspectivas sobre el artículo

Este artículo plantea una estrategia para reducir la velocidad de cómputo que implica diversas acciones, logrando un resultado satisfactorio. Tanto la aplicación del cómputo en paralelo como el preprocesamiento de las matrices involucradas contribuyen a que sean asequibles cómputos demasiado grandes a una mayor velocidad. Es una demostración clara de cómo existe un terreno fértil en el análisis de algoritmos para su paralelización.