Parallel One-Sided Block Jacobi SVD Algorithm: Resumen

Héctor Adolfo Corro Zárate[1]

Maestría en Ciencia de Datos
Instituto Tecnológico Autónomo de México

1. Introducción

En este artículo el autor describe en detalle una serie de ideas de acelerar, nombrar y trabajar con bloques de matrices en vez de elementos, el preprocesamiento de la matriz original, la inicialización especial del procedimiento, la nueva matriz recursiva y descomposición seno-coseno de ciertos bloques de matrices. La posible estrategia de paralelización para el algoritmo $one - sided\ block - Jacobi$ también es abordado.

Lo metódos jacobianos de un solo-lado para realizar la descomposición de valores singulares de una matriz son más eficientes que su contra parte de ambos-sentidos. Recientemente se ha visto que una buena implementación de un algoritmo jacobiano de un solo-lado de manera secuencial puede alcanzar la eficiencia de un método QR. Sin embargo, el QR utiliza una bidiagonalización como paso previo, lo que significa que le precisión relativa se pierde y no puede ser recuperada para un uso posterior.

Por ello, el autor se limita a reportar la generalización el algoritmo escalar jacobiano de un solo lada en bloques. Trabajar con bloques en computo secuencial conlleva a un mejor uso disponible de memoria, especialmente de cache, además de impulsar el flujo de los datos y su cálculo. Adicionalmente el computo en bloque permite un mayor grado de paralelización.

Principalmente se abordan 2 ideas, una primera abordada por Drmac y Vaselic para el jacobiano secuencial de un solo-lado y la otra propuesta por Oksa y Vajtersic para el jacobiano paralelo de ambos sentidos. Esto consiste en el preproceso de una matriz dada antes de llevarla a SVD al calcular su descomposición QR con columnas como pivote, de esta manera es aplicado el SVD al factor triangular. La segunda idea, que implementa la descomposición coseno-seno (CS) de cierta matriz

ortogonal en bloque lo cual lleva al tan llamado transformación de escala rápida en bloque ortogonal. Al final de las ideas el resultado es un uso más eficiente de memoria cache un reducido número de barridos requeridos para el algoritmo de convergencia a cierta precisión.

2. Algoritmos

Se presentan los siguientes algoritmos:

- **One-Sided Block-Jacobi Algorithm: OSBJA**: el OSBJA puede ser escrito como un proceso iterativo de la siguiente manera:

$$A^{(0)} = A, \qquad V^{(0)} = I_n A^{k+1} = A^{(k)} U^{(k)}, \qquad V^{(k+1)} = V^{(k)} U^{(k)}, \qquad k \geq 0 \tag{1}$$

  En esta parte se menciona que una de las estrategias de iteración más usadas son aquellas que se basan en el uso de las filas y columnas, donde el orden es implementado según lo dicho anteriormente.

- **Acelerando el OSBJA**: en esta parte se trata de mejorar la velocidad y eficiencia del OSBJA , el primer intento es reducir el número de barridas requeridas para la convergencia de todo el algoritmo a cierta precisión.

  En subtemas siguientes se abordan preprocesos que ayudan a reducir el número de iteraciones.

  - Factorizaciones QR y LQ
  - Inicializacón
  - Fast scaled block-orthogonal transformations

Algoritmo modificado: después de revisar todo lo anterior, se implementa en conjunto.

1. La priimer parte es calcular el producto de las matrices $Z = B_i^{(k)T} B_j^{(k)}$ donde (i, j) son la pareja actual del indice de pivote definido por el orden en bloque previo.

2. La segunda parte consiste en realizar los calculos en la memoria rápida o memoria Cache. Que en un momento requiere el calculo de $Q_{ii}^{(k)T} Z Q_{jj}^{(k)}$ y luego realizar la diagonalización de $\hat{A}_{ij}^{(k)}$

3. Calcular la factorización CS de la matriz eigenvector ortogonal de $\hat{U}^{(k)}$. Después de este paso se pueden estimar el número total de flops requeridos para esta parte del algoritmo.

En resumen, el preproceso de la matriz original A por la factorización QR con columnas pivote y la recursión aplicada a las 3 matrices pueden ser actualizadas de manera rápida en memoria (cache), mejorando y reduciendo el número de flops.

Criterio de paro: aquí es importante ver las cuestiones, menciona el autor, ya que si el proceso es parado muy temprano no se podría alcanzar la convergencia dada la precisión deseada. Si es parado muy tarde, iteraciones innecesarias podrían ponerse en marcha y se gastaría mucho más tiempo del necesitado.

3. Estrategia de paralelización y Conclusiones

En esta parte el autor decidió implementar OSBJA via MPI. Como el producto cruz de la matriz $\hat{A}_{ij}^{(k)}$ es calculado al principio de OSBJA usando dos bloques columna, para mantener la comunicación entre procesadores al mínimo, en este punto del calculo es normla asumir que el producto cruz de lo anterior es realizado en un procesador. Al principio asignan dos bloques consecutivos a dos procesadores consecutivos.

Una triste conclusión, desde el punto de vista del autor de este trabajo, es que en el artículo se menciona que no existe una convergencia global como resultado a ningún método paralelo para el Jacobiano en bloque.

Pese a todo, los elementos, técnicas y algoritmos propuestos sirven como base a futuras aproximaciones al caso aquí propuesto.