View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by El Servicio de Difusión de la Creación Intelectual

Una paralelización del método de Householder

Osvaldo Spositto, Gastón Procopio, Fabio Quintana, y Hugo Ryckeboer

Universidad Nacional de la Matanza
Departamento Ingeniería e Investigaciones Tecnológicas

Resumen La transformación de matrices a la forma de Hessemberg o triangular superior mediante el método de Householder es uno de los métodos más utilizados para obtener autovalores y autovectores de una matriz.
Cuando se intenta utilizarlo en un ambiente de multiprocesadores es conveniente que la matriz se pueda almacenar repartida en las memorias rápidas de los procesadores disponibles para contribuir al cálculo de las transformaciones $\left(I - \omega\,\omega^{\top}\right) A$ que por razones de eficiencia se calculan como $A - \omega\ \left(\omega^{\top} A\right)$. Distribuyendo a cada procesador los trozos de vector que se necesitan en cada operación (una contribución lineal), se alimenta una cantidad cuadrática de operaciones.
Cuando el tamaño de la matriz es tal que no cabe en la memoria se debe cargar y guardar sucesivos bloques de la misma en los procesadores disponibles, lo que baja su productividad por la cantidad cuadrática de ciclos de memoria lenta involucrados.
Se propone un replanteo del método de Householder que permite aprovechar la presencia de un bloque en la memoria para realizar múltiples tandas de actualización.

1. Introducción

Transformar los cómputos matriciales a esquemas de computación paralela se ha encarado hace tiempo [1,2]. Estos cambios se han enfocado de dos modos y en este orden: a) buscar operaciones del esquema de cálculo secuencial que se presten para un desarrollo paralelo. Los ciclos se prestan especialmente para estas transformaciones. b) Un segundo camino ha buscado un replanteo de los algoritmos de modo tal de poder expresar la meta por operaciones sobre grandes estructuras cada una de las cuales admite una paralelización ya conocida.

El primer enfoque se acerca mucho más a la arquitectura subyacente, para justificar y aumentar el ahorro de tiempo. El segundo enfoque se mantiene en un nivel más matemático ya que la paralelización la consigue recurriendo a subprogramas ya optimizados programados en todos sus detalles por los proveedores de ambientes y bibliotecas de cálculo paralelo como ser el MATLAB, ScaLAPACK, CUBLAS, OPENCL, etc. Se tarda meses en emular la eficiencia y flexibilidad de adaptación a una diversidad de estructuras que estos productos han conseguido.

Los métodos de alto nivel se pueden encuadrar bajo un denominador común: la división en bloques.

La transformación de esquemas de cálculo pensados para escalares introduce inevitablemente muchos productos entre bloques y sus derivados, que es una de las operaciones mejor implantadas en arquitecturas paralelas. A las divisiones entre escalares le suelen corresponder inversiones de bloques. Aunque los tamaños de los mismos pueden ser variables, los elementos de la diagonal principal ocupan bloques cuadrados. Rotella y Zambettakis [3] lograron adaptar el método de Householder a una descomposición en bloques.

Anderson, Ballard, Demmel, Keutzer [4] realizan un desarrollo orientado a matrices de muchas filas y pocas columnas, mostrando la posibilidad de lograr un uso intensivo de cálculo en cada procesador.

El método aquí propuesto está orientado a matrices cuadradas de gran tamaño, usuales en el método de recuperación de información LSI. Procede por etapas. La cantidad de etapas depende de la relación entre el tamaño de la matriz y la memoria total disponible entre todos los procesadores. Analíticamente se basa en una relectura de la transformación de Householder y tiene en vista que nunca deba ser cargado durante una etapa, en la memoria de registros de los procesadores, un mismo elemento de la matriz. Como la cantidad de operaciones es igual a la del método original salvo términos de menor orden, resulta que el esquema propuesto mantiene, a diferencia de una implantación directa de aquel, una muy alta relación entre cómputos y transferencias.

Teniendo en cuenta lo indicado, el método ocupa respecto de la clasificación inicial una posición intermedia.

2. Revisión de la transformación de Householder

La transformación de Householder permite transformar un vector en otro, con cierta limitación, por una reflexión respecto de un plano. Householder al plantearlo tenía en mente anular los elementos por debajo de una diagonal.

La transformación se puede describir con una matriz unitaria, la cual, por su número de condición unitaria, mantiene baja la propagación de los errores del cálculo.

La reflexión es una operación que no se limita a las necesidades del método tal como lo usa Householder.

En el planteo habitual se aprovecha para transformar un vector columna en un vector de ceros por debajo de la primera diagonal o subdiagonal.

Sin embargo esto no es una exigencia de la transformación sino de una estrategia para lograr en pocos pasos la deseada forma ya sea triangular superior o de Hessemberg.

Supongamos tener un vector u que se quisiera transformar por reflexión en un vector v. La única exigencia para que esto sea posible es que v tenga el mismo tamaño. De aquí surge una primera ecuación $||u||_2 = ||v||_2$.

Hablar de tamaño con sentido geométrico no es más que tomar normas euclídeas. Cuando el método se aplica con la intención de conseguir ciertos valores en determinados elementos es necesario disponer de otro, un comodín, al que se le hará asumir el valor necesario para igualar módulos. Si los valores nuevos

superan a los viejos, el objetivo se hace imposible. En el uso que propuso Householder de esta transformación, que es la de anular elementos, ello es siempre posible. También lo será en esta propuesta. Tenemos aquí una primera ecuación: $||u||_2 = ||v||_2$ que contiene una incógnita: el nuevo valor del comodín S, pudiendo elegir su signo.

Utilizando el método para triangular una matriz, el comodín pertenece a la diagonal y si es para llegar a la forma de Hessemberg el comodín pertenece a la subdiagonal.

No se pretende repetir íntegra la deducción que está reproducida en textos de cálculo numérico [5,6], sino llamar la atención a las características a tener en cuenta para comprender la modificación propuesta:

- El término comodín se sustituye por el largo del subvector armado por su antiguo valor y los términos que se quieren anular. Esta operación, una distancia euclídea, culmina con una raíz cuadrada cuyo signo se puede elegir según la conveniencia de la estabilidad numérica.
- El vector ω tiene componentes nulos en correspondencia a los elementos de u que no cambian.
- Los elementos que se anulan con la transformación ingresan tal cual en $u - v$.
- El comodín es el único que aporta una real diferencia. Como hay interés en tener un valor elevado se elige el signo de la raíz cuadrada como opuesto al signo del antiguo valor para que haya una suma de valores absolutos en lugar de una diferencia.
- Finalmente para tener ω es necesario escalar el vector diferencia de modo tal que su módulo sea 1 para satisfacer $\omega^\top \omega = 1$.

La aplicación de la transformación a la matriz A modifica todas sus columnas. Eligiendo el orden de los productos indicados se obtiene un esquema eficiente de cálculo:

$$(I - 2\omega\omega^\top)\, A = A - 2\omega\, (\omega^\top A) \quad . \tag{1}$$

Al producto $\omega^\top A$ en esta ecuación lo designaremos $\widetilde{\omega}^\top$. Si el producto externo $2\omega\widetilde{\omega}^\top$ se calcula por columnas no es necesario tener la totalidad de $\widetilde{\omega}^\top$ calculado, sino que el cálculo de este vector puede avanzar conforme se lo necesita. Esto permite procesar matrices donde la cantidad de columnas es tal que no se puedan tener íntegras en los procesadores.

En el método propuesto el vector v tendrá algunos ceros más que u, consiguiendo en sucesivas etapas la anulación del triángulo inferior deseado.

Un desarrollo similar se efectúa en sucesivas etapas sobre las columnas siguientes.

El desarrollo propuesto está orientado a matrices cuyo tamaño impide tenerlas íntegras en las memorias rápidas de los procesadores disponibles.

Si se aplicara la fórmula 1 a estas matrices se cargaría bloques de A y un correspondiente subvector de $\omega^\top$, con ello se puede computar la contribución del bloque al producto de la derecha y luego, esperar hasta tener el producto completado entre los diversos procesadores que contribuyen con las columnas del

bloque considerado. Luego cargar el subvector correspondiente al bloque y efectuar el producto externo con ω para modificar a A. A continuación almacenar el bloque en la memoria grande para poder procesar a otro bloque. Las transferencias que son lentas suman: 2 veces el bloque completo: carga y descarga, y 4 de tamaño lineal. La operaciones flotantes dos veces el tamaño del bloque, lo cual constituye un esquema de baja eficiencia.

3. Subdivisión de la matriz en franjas para su proceso

Se presupone una arquitectura en la cual cada procesador dispone de una memoria exclusiva que además de rápida es el lugar en el cual deben residir los operandos. La matriz dato y resultado residen en una memoria compartida la cual es también el vínculo para intercambiar mensajes. Finalmente se supone que la matriz dato es demasiada voluminosa para tenerla íntegra en la memorias rápidas. Un algoritmo para ser eficiente debe realizar gran número de operaciones con los elementos almacenados en su memoria rápida antes de sustituirlos por nuevos datos. Los mensajes a intercambiar son los vectores ω. La suma de todos ellos es una fracción del espacio que ocupa la matriz.

El algoritmo procesará por etapas y en cada etapa hará un nuevo reparto de la matriz A de $n \times n$ a transformar.

Para poder hacer un uso intensivo de partes de la matriz, se la supone dividida en franjas horizontales. En cada subetapa se tendrá en memoria la primera y una de las siguientes.

La primera con p filas, las siguientes con q, la última posiblemente con fracción de q. Los tamaños p y q, de ser conveniente se pueden redefinir en cada etapa, pero están influenciados por los recursos disponibles.

Los elementos de cada franja se almacenarán en sendos procesadores, si por insuficiente número de procesadores no se pudieran almacenar las franjas de una subetapa completas, bastaría con tener completas un grupo de columnas y una vez procesadas estas pasar al grupo siguiente.

Si el objetivo es triangular, la primera franja abarcará las filas 1 a p. Si es llegar a la forma de Hessemberg, como la primera fila no cambia, tomará de la 2 a la $p+1$. Cada etapa quedará dividida en subetapas. Una subetapa corresponde al proceso de una de las franjas de tamaño q.

Para unificar ambos objetivos de aquí en más la numeración de filas se referirá a lo almacenado.

3.1. Primera subetapa

La primera subetapa agrupará la franja de ancho p y la primera de tamaño q, ambas instaladas en las memorias de sendos microprocesadores, manteniendo en procesadores independientes los elementos de ambas franjas.

Procedemos a aplicar sobre las columnas 1 a p la transformación de Householder anulando hasta la fila $p + q$.

Las operaciones a efectuar serán descritas para una columna genérica $k \in 1 \ldots p$. Se trata de anular en esa columna los elementos de la filas $k+1 \ldots p+q$. El cálculo a efectuar para determinar ω es idéntico al que se efectuaría si se tratara de una matriz de tamaño $p+q$.

El vector ω paralelo a $u-v$ tiene ceros donde esta diferencia es cero, los que se corresponden a los elementos que no sufren cambios que son las posiciones $1 \ldots k-1$ y $p+q+1 \ldots n$. Los elementos no nulos están en los lugares $k \ldots p+q$. El elemento k es el que oficia de comodín y mantiene la igualdad de módulos.

El siguiente paso es analizar que sucede en el cálculo de $A - 2\omega(\omega^\top A)$.

La fila $\omega^\top$ debe construir un producto interno con las sucesivas columnas de A. Los elementos de la columna a los que corresponden ceros en el primer factor pueden ser ignorados. O sea, de cada columna de A sólo necesitamos acceso a las filas $k \ldots p+q$ y estas pertenecen a las dos franjas almacenadas. El resultado de estos sucesivos productos internos es un vector fila $\widetilde{\omega}^\top$, posiblemente completo a partir del lugar k. Como las columnas se procesan en orden, en las columnas anteriores a k ya hay ceros y se puede omitir un cálculo que daría cero.

Si la distribución de elementos de A por procesador se hizo por bloques, recibiendo una cantidad lineal de datos se organiza un cálculo cuadrático y una emisión lineal de resultados.

Más adelante se analiza las dimensiones ideales de tales bloques. En el último paso el vector columna ω multiplica al vector fila recién obtenido $\widetilde{\omega}^\top$.

Cada elemento de A se actualiza con el esquema $a_{ij} = a_{ij} - 2\omega_i\widetilde{\omega}_j$. En la filas donde ω_i se anula no hay nada que procesar, sufriendo cambios solamente las filas almacenadas en la memoria. Este cálculo exige distribuir a cada procesador los subvectores de ω y de $\widetilde{\omega}$ que necesita para su actualización local. Trabajo que con una transferencia lineal organiza también un cálculo de volumen cuadrático.

Un mismo procesador dedicado a contener una submatriz de la franja realiza estos dos trabajos cuadráticos p veces lo que justifica el trabajo cuadrático de su carga y descarga.

3.2. Siguientes subetapas

Se conserva el contenido de la primera franja y se sustituye la segunda franja con las siguientes q filas de la matriz A. Para no complicar las fórmulas se supone en lo que sigue que esta es la tercera subetapa, segunda después de la inicial.

El propósito es anular ahora los elementos del bloque determinado por las filas $p+2q+1 \ldots p+3q$ y columnas $1 \ldots p$.

Nuevamente se procesa en orden las columnas. Para la columna k la situación y proceso es el siguiente:

- A las filas $1 \ldots k-1$ pertenecientes al triángulo superior no hay interés en cambiarlas
- La fila k volverá a cumplir el papel de comodín para igualar normas.
- Ya tiene ceros en las filas $k+1 \ldots p+2q$ y no se quiere cambiarlas.
- Las filas $p+2q+1 \ldots p+3q$ deben anularse.
- Las filas $p+3q+1 \ldots n$ no sufren cambios.

El vector ω de esta subetapa, construido por diferencia entre el antes y el después tiene elementos no nulos solamente en la posiciones k y $p+2q+1 \ldots p+3q$. Al efectuar el producto $\omega^\top A$ solamente intervienen estas filas de A. La k, solitaria, ubicada en la primera franja; las otras en la segunda franja de la memoria que en este momento contiene la cuarta franja de A.

Ese producto genera un vector fila casi completo desde el lugar k y ceros a su izquierda, porque ya hay ceros en la columnas anteriores.

La modificación final de los elementos de A sigue el mismo esquema discutido para la primera subetapa.

Los elementos de la primera franja trabajan menos, un elemento para cada transformación de A, para cada k uno distinto con lo cual todos intervienen al menos una vez por subetapa.

Lo mismo sucede en el producto externo que actualización a A.

3.3. Etapas posteriores

Al finalizar la primera etapa la matriz se encuentra en el mismo estado en que hubiera estado después de p etapas del método de Householder. Las primeras p columnas y p filas de A no vuelven a intervenir y ya tienen sus valores defintivos. Luego hay que procesar la submatriz restante, cuadrada de tamaño $n-p$, lo que requiere un replanteo de los tamaños p y q.

4. Reflexiones sobre los tamaños

Hay varios tamaños por discutir: los parámetros p y q y la configuración de cada procesador. Estos elmentos interactúan, pero conviene entender a cada uno por separado.

4.1. La elección de p y q

Hay que tener presente que múltiples características de la arquitectura de proceso disponible influyen en la elección de los parámetros. Las variables a concretar son discretas de modo tal que modelos continuos sólo sirven de orientación en el diseño.

La precisión se ve afectada si las franjas son demasiado pequeñas. Las publicaciones citadas omiten comentarios sobre la incidencia de los esquemas que proponen en la precisión. La experiencia numérica indica que la cantidad de franjas de tamaño q no debiera pasar de 10. Si se conserva los S^2 de cada etapa el valor del elemento a_{kk} al final de una etapa coincide con el valor del método original.

El volumen de cálculo sólo se ve afectado en un término menor. Para efectuar una transformación es necesario un cálculo preparatorio, obtener ω el cual depende de dos normas. Pero en el método estandar estas normas se calculan sobre vectores del orden de $n-k$ y en este reordenamiento, en la primera subetapa

con vectores de tamaño $p+q-k$, y en las siguientes subetapas con vectores de tamaño q con lo cual el trabajo total es comparable con el del método directo. Sólo está el trabajo extra de reinicialización de ciclos en cada subetapa, que frente a otras magnitudes parece ser despreciable.

Cada reducción genera una matriz ortogonal $H = I - \omega\omega^\top$, habiendo r reducciones por columna en lugar de una, su cantidad aumenta r veces. Conceptualmente, su producto es la matriz ortogonal que junto con la triangular superior podrían reconstruir la matriz original A. Hay que tener presente que los vectores ω que los engendran tienen $q+1$ elementos no nulos, a excepción de la primera que tiene $p+q-k$, de modo tal que multiplicar por ellas cuesta r veces menos. que operar con las ω extensos del método original. Almacenar los ω correspondientes a un mismo valor de k para postergar estos productos insume $p+q-k$ para la primera subetapa y $q+1$ para cada una de las $r-1$ posteriores, su suma $p+q-k+(r-1)(q+1) = p+rq-k+r-1 = n-k+r-1$ contra $n-k$ de un ω grande del planteo original de Householder evidencia que insume sólo $r-1$ lugares extras.

Cada transformación afecta a una fila de la primera franja y a todas las filas de la que esté ocupando la segunda franja presente. Cuando se procesa nuevamente para un mismo valor de k la segunda franja en memoria ya no es la misma. Si en total la matriz A fue descompuesta en una franja de ancho p y r de ancho q, la primera subetapa opera por cada valor de k sobre $p+q-k$ filas y las siguientes sobre $q+1$, en total se procesan $p+q-k+(r-1)(q+1) = p+rq-k+r-1 = n-k+r-1$ contra $n-k$ del planteoo original, o sea $r-1$ filas extras. El aumento del proceso se corresponde con el aumento de almacenamiento y en ambos casos representa un término de menor orden.

Beneficio de un q pequeño. Mantener q bajo permite aumentar el valor de p, lo que implica menos etapas y por consiguiente menos lecturas en proporción al cálculo.

Inconveniente de un q demasiado pequeño. Pero, un q bajo implica que los procesadores que contienen bloques, submatrices de la matriz a factorear, tienen pocas filas y consiguientemente más columnas. Una gran desproporción entre filas y columnas baja el rendimiento del procesador, expresado como cociente entre flops y transferencias.

4.2. Dimensionamiento óptimo de un bloque

Se puede tomar como indicador la relación entre volumen de las lecturas y volumen del cálculo, cuanto más bajo, mejor. Simplifica los análisis posteriores analizar primero las dimensiones óptimas de un bloque.

El cálculo se mide en flops, la operación típica del álgebra lineal, que abarca un producto y suma de reales.

Forma óptima de un bloque. Un bloque, o submatriz, se define por una cantidad a de filas y una cantidad b de columnas. Estas cantidades no son independientes sino que su producto es el tamaño B que ocupa el bloque en la

memoria, una fracción de la memoria rápida M de la cual dispone el procesador. Fijado B sabemos que esto dará lugar a un volúmen $2B$ de transferencias.

Por cada valor de k se repite el mismo proceso: lectura de un vector columna de tamaño a y varias acciones de transferencia de vectores fila de tamaño b. Como esta cantidad dependerá de decisiones posteriores se parametrizará con t.

El volumen de transferencias que se hacen para cada valor de k es: $a+tB/a$, expresión que se minimiza con $a = \sqrt{tB}$ y de lo cual se deduce $b = \sqrt{B/t}$. La cantidad total de transferencias con formato óptimo resulta ser $2\sqrt{tB}$.

Primer diseño de *q*. Para minimizar la comuncación entre bloques es conveniente que las q filas ocupen un sólo bloque, lo que equivale a decir $a = q$. Cada procesador asignado a la franja se ocupa exclusivamente de una cantidad b de columnas.

Cada bloque (salvo los que contienen las primeras p columnas) interviene en las siguientes operaciones:

Carga del bloque Son ab lecturas.

Elaboración de $\widetilde{\omega}^\top$ El procesador debe recibir a elementos y efectúa ab flops con lo cual construye un vector de tamaño b. Éste se debe combinar con el cómputo de la fila k. Lo puede hacer de dos formas, envía su vector al procesador que tenga la fila k y recibe la suma terminada o recibe la contribución de la fila k, realiza la suma y la devuelve. En ambos casos son dos transferencias de tamaño b y puede emprender el siguiente cómputo.

Actualización de los elementos de A que están en el bloque La cantidad de operaciones es nuevamente ab

Grabación del bloque Son ab grabaciones.

Gracias al tratamiento propuesto la segunda y tercera operación se realizan p veces. La relación de a y b ya fue discutida y del análisis surge que la cantidad de transferencias t de dimensión b que se efectúa es 3. Lo que permite establecer:

$$\text{Relación de E/S a Flops} = \frac{2p\sqrt{3B} + 2B}{2pB} = 1/p + \sqrt{3/B}\ . \qquad (2)$$

que señala la conveniencia de p y B grandes. El tamaño de B será una porción de la memoria M que la arquitectura provee a cada procesador, después de restar el espacio para las transferencias de vectores de tamaños a y b hacia la memoria compartida.

Compensación del desbalanceo de la carga Hasta aquí se ha ignorado a los procesadores que contienen los elementos de las primeras p filas. Estos computan de un modo similar pero en lugar de recibir un factor izquierdo de a elementos sólo reciben uno efectuando en total $2b$ flops y $3b + 1$ transferencias por cada k. Siendo sus transferencias comparables con la de un procesador de la franja inferior, no tiene retardo por ello, pero tiene capacidad de cálculo ociosa.

De aquí surge la conveniencia de estudiar si un sistema mixto no ofrece mejor productividad. Teniendo $p >> q$ como el cálculo precedente ha sugerido tenemos muchos procesadores con baja ocupación. Sea f un coeficiente que exprese la relación que expresa $p/q + 1$. Por cada procesador activo al máximo hay $f - 1$ poco cargados.

Tratando de tener una carga homogénea se puede redistribuir la carga de modo tal de que se almacene en un procesador por cada fila de la segunda franja $f-1$ de la primera. Para simplificar las cuentas y tener un esquema homogéneo de trabajo supondremos f una potencia de dos.

El tamaño del bloque se ve reducido a $B = M/f$. La cantidad de procesador que contienen las mismas columnas es f. El cálculo de $\widetilde{\omega}^{\top}$ exige integrar en una única suma los aportes de cada uno lo que exige $2\log_2 f$ intercambios de mensajes. Lo que sí se ha ahorrado es la comunicación por cada valor de k del aporte de la fila k; alguno de los procesadores la tiene. Esto da $t = 1+2\log_2 f$. El tamaño óptimo de a en cada procesador es $a = \sqrt{tB} = \sqrt{(1+2\log_2 f)B/f}$. El volumen total de transferencias en cada procesador es $2p\sqrt{(1+2\log_2 f)B/f} + 2B/f$ y el cálculo ocupa $2pB/f$.

En el tiempo en que trabaja un procesador para procesar un bloque de tamaño B/f trabajan f procesadores y completan el proceso de B el bloque que inicialmente hubiera estado solo en un procesador.

El inconveniente del reparto es el aumento de $q = fa$, y fija un valor de $p = (f-1)q = f(f-1)a$.

Para poder comparar las distintas soluciones, cuyas unidades de procesamiento no son iguales, se propone tomar el esfuerzo de procesar un bloque de tamaño B sometido a una reducción. Lo que da en transferencias $2\sqrt{(1+2\log_2 f)B/f} + 2M/\left(f^2(f-1)\right)$.

El cuadro 1 muestra los valores de las variables en este esquema de carga distribuida. La parte inferior del cuadro estudia el comportamiento si se mantienen la primera y segunda franja separadas en dos grupos de procesadores. Se observa que resulta ventajoso el esquema de ubicación compartida ya que aprovecha con una carga más homogénea la capacidad de cálculo de todos los procesadores.

Por otra parte bajan las transferencias y cómputos que realiza cada procesador en paralelo al aumentar su número.

Para cantidades de procesadores que no sean las tabuladas hay que diseñarlo a medida porque los procesadores no intercambian la misma cantidad de mensajes.

También es legítimo apartarse ligeramente de los valores de a y b sugeridos ya que cerca del óptimo los cambios tienen poca incidencia.

Los procesadores asignados a las primeras p columnas tienen menos trabajo ya que no necesitan calcular los ceros que la teoria asegura que producen.

5. Experimento

Se ha supuesto que cada una de 4 unidades aritméticas tuviera 128K de memoria de trabajo y que, despues de planear su uso se tuviera 100K disponibles lo que equivale a 12.500 números reales.

Con estos valores, aplicando las fórmulas de la segunda columna $p = 1644$, $q = 548$. Para que el proceso fuera por franjas plenas se construyó con azar una matriz simétrica con $N = 3288$. Se la trianguló en la forma habitual y siguiendo el esquema propuesto. Todo el cálculo se hizo en doble precisión. El error medio

Cuadro 1. Valores de los parámetros para diversas configuraciones

Procesadores asignados	unidad	2	4	8	16
Fracción de memoria franja 2		0,500	0,250	0,125	0,063
Transferencias de dimensión a		1	1	1	1
Transferencias de dimensión b		3	5	7	9
a óptimo	$\sqrt{B}$	1,225	1,118	0,935	0,750
b óptimo	$\sqrt{B}$	0,408	0,224	0,134	0,083
Ancho de la 2da franja	$\sqrt{B}$	2,449	4,472	7,483	12,000
Ancho de la 1ra franja	$\sqrt{B}$	2,449	13,416	52,383	180,000
Transferencias de vectores	$\sqrt{B}$	2,449	2,236	1,871	1,500
Transferencias carga y descarga	$\sqrt{B}$	0,408	0,037	0,005	0,001
Flops	B	1	0,5	0,25	0,125
Relación entre E/S y proceso	$\sqrt{1/B}$	2,858	4,547	7,502	12,006
Ancho de la 2da franja	$\sqrt{B}$	1,732	1,732	1,732	1,732
Ancho de la 1ra franja	$\sqrt{B}$	1,732	5,196	12,124	25,981
Transferencias de vectores	$\sqrt{B}$	3,464	3,464	3,464	3,464
Transferencias carga y descarga	$\sqrt{B}$	1,155	0,385	0,165	0,077
Flops	B	2	2	2	2
Relación entre E/S y proceso	$\sqrt{1/B}$	2,309	1,925	1,815	1,771

cuadrático fue $1{,}2740e-12$ con un máximo de $1{,}1642e-10$, ubicado en la primera fila.

Referencias

1. Dianne P. O'leary, G. W. Stewart Data-flow algorithms for parallel matrix computations Comm. ACM 28, 1985, 840-853.
2. Dianne P. O'leary, G. W. Stewart: Assignment and scheduling in parallel matrix factorizations, Linear Algebra Appl. 77, 1986, 275-299.
3. F. Rotella, F., Zambettakis Block Householder transformation for parallel QR factorisation. Appl. Math. Lett. Vol 12, Issue 4, May 1999, 29-34
4. Communication-Avoiding QR Decomposition for GPUs Michael Anderson, Grey Ballard, James Demmel, Kurt Keutzer Technical Report No. UCB/EECS-2010-131, University of California at Berkeley,October 8, 2010 http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-131.html
5. Gene H. Golub, Charles F. Van Loani Matrix Computations 4th Edition, Johns Hopkins University Press, 2013
6. William H. Press, Saul A. Toukolsky, William T. Vetterling, Brian P. Flannery Block Householder transformation for parallel QR factorisation. Appl. Math. Lett. Vol 12, Issue 4, May 1999, 29-34