Estrategias para Multiplicación de Matrices

Nancy Hitschfeld Kahler

Departamento Ciencias de la Computación, Universidad de Chile, Santiago, Chile
(Basado en el tutorial de Tim Mattson, Intel, 2016)
(Incluido en Lecturas Complementarias u-cursos)

April 14, 2024

Outline

Multiplicación de Matrices

- Composición de dos transformaciones lineales en ese espacio vectorial

Multiplicación de Matrices

- Composición de dos transformaciones lineales en ese espacio vectorial
- Es clave en la resolución de problemas en algebra lineal

Multiplicación de Matrices

- Composición de dos transformaciones lineales en ese espacio vectorial
- Es clave en la resolución de problemas en algebra lineal
- Objetivo: Calcular $C = A * B$, donde:
  - $C \in R^{N \times M}$

Multiplicación de Matrices

- Composición de dos transformaciones lineales en ese espacio vectorial
- Es clave en la resolución de problemas en algebra lineal
- Objetivo: Calcular $C = A * B$, donde:
  - $C \in R^{N \times M}$
  - $A \in R^{N \times P}$

Multiplicación de Matrices

- Composición de dos transformaciones lineales en ese espacio vectorial
- Es clave en la resolución de problemas en algebra lineal
- Objetivo: Calcular $C = A * B$, donde:
  - $C \in R^{N \times M}$
  - $A \in R^{N \times P}$
  - $B \in R^{P \times M}$
- Objetivo: explorar distintas estrategias y evaluar cuál funciona mejor.
- Cómo debería ser la experimentación? Qué estrategias probar? Cómo compararlas?
- Idea: construir distintas opciones y evaluarlas en la arquitectura disponible.

Algoritmo secuencial

```
void multMat(int Ndim, int Mdim, intPdim, float *A, float *B, float *C)
{
    int i, j, k;
    for(i=0; i<Ndim; i++){
        for(j=0; j<Mdim; j++){
            for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
                C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];

            }
        }
    }
}
```

- Es importante contar con una implementación del algoritmo secuencial sin optimización
- Plataforma 1: Apple laptop: gpu GeForce® 8600M GPU from NVIDIA with a max of 4 cores. CPU is Intel® Core™2 Duo CPU T8300 @ 2.40GHz.
- Plataforma 2: Device is Intel® Xeon® CPU, E5649 @ 2.53GHz. Device is Tesla® M2090 GPU from NVIDIA® with a max of 16 compute units, 512 PEs
- Se comparará desempeño en los cores de la CPU y en la GPU.

Cómo transformarlo en paralelo? OpenCL

```
void multMat(int Ndim, int Mdim, int Pdim, float *A, float *B, float *C)
{
    int i, j, k;
    for(i=0; i<Ndim; i++){
        for(j=0; j<Mdim; j++){
            for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
                C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
            }
        }
    }
}
```

- Paso 1: Marcar como kernel y especificar tipo de parámetros

Cómo transformarlo en paralelo? OpenCL

```
void multMat(int Ndim, int Mdim, int Pdim, float *A, float *B, float *C)
{
    int i, j, k;
    for(i=0; i<Ndim; i++){
        for(j=0; j<Mdim; j++){
            for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
                C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
            }
        }
    }
}
```

- Paso 1: Marcar como kernel y especificar tipo de parámetros
- __kernel multMat(const int Ndim, const int Mdim, const int Pdim, __global float *A, __global float *B, __global float *C)

Cómo transformarlo en paralelo? OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global float *B, __global float *C)
{
    int i, j, k;
    for(i=0; i<Ndim; i++){
        for(j=0; j<Mdim; j++){
            for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
                C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
            }
        }
    }
}
```

- Paso 2: Eliminar las intrucciones for y asignar índices de los work items

Cómo transformarlo en paralelo? OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global float *B, __global float *C)
{
    int i, j, k;
    for(i=0; i<Ndim; i++){
        for(j=0; j<Mdim; j++){
            for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
                C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
            }
        }
    }
}
```

- Paso 2: Eliminar las intrucciones for y asignar índices de los work items
- i= get_global_id(0);j = get_global_id(1);

Cómo transformarlo en paralelo? OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global  float *B, __global float *C)
{
    int i, j, k;
    i= get_global_id(0);
    j = get_global_id(1);
    for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
    }
}
```

- Paso 3: Qué optimización local podríamos hacer?

Cómo transformarlo en paralelo? OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global  float *B, __global float *C)
{
    int i, j, k;
    i= get_global_id(0);
    j = get_global_id(1);
    for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        C[i*Mdim+j] += A[i*Pdim+k] * B[k*Mdim+j];
    }
}
```

- Paso 3: Qué optimización local podríamos hacer?
- Servirá definir una variable local?

Cómo transformarlo en paralelo? OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global float *B, __global float *C)
{
    int k;
    int i= get_global_id(0);
    int j = get_global_id(1);

    float suma=0;
    for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        suma += A[i*Pdim+k] * B[k*Mdim+j];
    }
    C[i*Mdim+j] = suma;
}
```

Estrategias MFlops	Plataforma 1		Plataforma 2	
	CPU	GPU	CPU	GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9

Cómo optimizar el código paralelo OpenCL

- De qué orden computacional es la multiplicación de matrices?

Cómo optimizar el código paralelo OpenCL

- De qué orden computacional es la multiplicación de matrices?
  - $2 * n^3$ operaciones punto flotante (flops)

Cómo optimizar el código paralelo OpenCL

- De qué orden computacional es la multiplicación de matrices?
  - $2 * n^3$ operaciones punto flotante (flops)
  - Número total de valores $3 * n^2$

Cómo optimizar el código paralelo OpenCL

- De qué orden computacional es la multiplicación de matrices?
  - $2 * n^3$ operaciones punto flotante (flops)
  - Número total de valores $3 * n^2$
- Para optimizar, con cada movimiento/copia de memoria, debemos hacer el mayor número de operaciones posible

Cómo optimizar el código paralelo OpenCL

- De qué orden computacional es la multiplicación de matrices?
  - $2 * n^3$ operaciones punto flotante (flops)
  - Número total de valores $3 * n^2$
- Para optimizar, con cada movimiento/copia de memoria, debemos hacer el mayor número de operaciones posible
- Podemos acelerar el cálculo del producto punto usando work groups y memoria compartida? Cómo?

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)
- Usar grupos de 128x128 (dimension local)

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)
- Usar grupos de 128x128 (dimension local)
  - Sincronización entre work items, dentro de un grupo, a través de barriers y memory fences

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)
- Usar grupos de 128x128 (dimension local)
  - Sincronización entre work items, dentro de un grupo, a través de barriers y memory fences
  - No se puede sincronizar fuera de un grupo

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)
- Usar grupos de 128x128 (dimension local)
  - Sincronización entre work items, dentro de un grupo, a través de barriers y memory fences
  - No se puede sincronizar fuera de un grupo
- Importante: Escoger la mejor dimensión para tu problema.

Cómo optimizar el código paralelo OpenCL

- Usar una matriz de work items. Por ejemplo 1024x1024 (dimension global)
- Usar grupos de 128x128 (dimension local)
  - Sincronización entre work items, dentro de un grupo, a través de barriers y memory fences
  - No se puede sincronizar fuera de un grupo
- Importante: Escoger la mejor dimensión para tu problema.

Modelo de memoria en OpenCl

- Memoria privada — por work item

Modelo de memoria en OpenCl

- Memoria privada — por work item
- Memoria Local — compartida dentro de un grupo

Modelo de memoria en OpenCl

- Memoria privada — por work item
- Memoria Local — compartida dentro de un grupo
- Memoria global y constante — visible por todo los grupos

Modelo de memoria en OpenCl

- Memoria privada — por work item
- Memoria Local — compartida dentro de un grupo
- Memoria global y constante — visible por todo los grupos
- Memoria del host — solo acceso desde la CPU

Modelo de memoria en OpenCl

- Memoria privada — por work item
- Memoria Local — compartida dentro de un grupo
- Memoria global y constante — visible por todo los grupos
- Memoria del host — solo acceso desde la CPU
- Manejo de memoria es explícito

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando
- En la arquitectura simple de apple usada:

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando
- En la arquitectura simple de apple usada:
  - Espacio del problema 1000x1000

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando
- En la arquitectura simple de apple usada:
  - Espacio del problema 1000x1000
  - Dimension Global 1000 (1 fila por work item)

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando
- En la arquitectura simple de apple usada:
  - Espacio del problema 1000x1000
  - Dimension Global 1000 (1 fila por work item)
  - Dimension Local 250 (un work group por core de la GPU)

Cómo optimizar el código paralelo OpenCL

- Qué pasa si usamos un solo work item calcular toda una fila C(i,...)?
  - El cálculo de una fila siempre usa la misma fila de A
  - Las columnas de B van variando
- En la arquitectura simple de apple usada:
  - Espacio del problema 1000x1000
  - Dimension Global 1000 (1 fila por work item)
  - Dimension Local 250 (un work group por core de la GPU)

Cómo optimizar el código paralelo OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global float *B, __global float *C)
{
    int k;
    int i= get_global_id(0);

    float suma;

    for(j=0; j<Mdim; j++){
        suma = 0;
        for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        suma += A[i*Pdim+k] * B[k*Mdim+j];
        }
        C[i*Mdim+j] = suma;
   }
}
```

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8

Cómo optimizar el código paralelo OpenCL

- Ayuda o no ayuda esto? Qué pasará

Cómo optimizar el código paralelo OpenCL

- Ayuda o no ayuda esto? Qué pasará
- Cómo se podrá hacer mejor?

Cómo optimizar el código paralelo OpenCL

- Ayuda o no ayuda esto? Qué pasará
- Cómo se podrá hacer mejor?
- Usar memoria privada del work item para almacenar fila i

Cómo optimizar el código paralelo OpenCL

- Ayuda o no ayuda esto? Qué pasará
- Cómo se podrá hacer mejor?
- Usar memoria privada del work item para almacenar fila i

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global  float *B, __global float *C)
{
    int k,j;
    int i= get_global_id(0);
    float Awrk[1000];

    for(k=0;k<Pdim;k++) Awrk[k] = A[i*Pdim+k];

    float suma;
    for(j=0; j<Mdim; j++){
        suma = 0;
        for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        suma += Awrk[k] * B[k*Mdim+j];
        }
        C[i*Mdim+j] = suma;
   }
}
```

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3

- Impacto grande!!

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3

- Impacto grande!!
- Se podrá hacer mejor?

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3

- Impacto grande!!
- Se podrá hacer mejor?
- Almacenar columnas de B en memoria local por cada work group.
- Nota: Dentro de cada grupo, los work items usan la misma columna B en algún momento

Cómo optimizar el código paralelo OpenCL

```
__kernel multMat(const int Ndim, const int Mdim, const int Pdim,
                 __global float *A, __global  float *B, __global float *C,
                 __local float* Bwrk))
{
    int k,j;
    int i= get_global_id(0);
    int iloc= get_local_id(0);  // indice del work item dentro del grupo
    int nloc= get_local_size(0); // tamano grupo

    float Awrk[1000];

    for(k=0;k<Pdim;k++) Awrk[k] = A[i*Pdim+k];

    float suma;
    for(j=0; j<Mdim; j++){
        for(k=iloc;k<Pdim;k=k+nloc)
            Bwrk[k] = B[k*Mdim+j];
        barrier(CLK_LOCAL_MEM_FENCE); // la columna B esta completa

        suma = 0;
        for(k=0;k<Pdim;k++){ //C(i,j) = sum(over k) A(i,k) * B(k,j)
        suma += Awrk[k] * B[k*Mdim+j];
        }
        C[i*Mdim+j] = suma;
    }
}
```

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3
C row per work-item, A row private B row local		2472	10.047,5	8.121,9

- Correr tests en vuestro sistema!!!
- Obtener la mejor optimizacion de la CPU

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3
C row per work-item, A row private B row local		2472	10.047,5	8.121,9

- Correr tests en vuestro sistema!!!
- Obtener la mejor optimizacion de la CPU
  - Usar optimización del compilador

Cómo optimizar el código paralelo OpenCL

Estrategias MFlops	Plataforma 1 CPU	Plataforma 1 GPU	Plataforma 2 CPU	Plataforma 2 GPU
Secuencial (not opencl)	167	N/A	887,2	N/A
C(i,j) por work item, solo memoria global	744	511	3.926,1	3.720,9
C row per work-item, all global		258	3.379,5	4.195,8
C row per work-item, A row private		873	3.385,8	8.584,3
C row per work-item, A row private B row local		2472	10.047,5	8.121,9

- Correr tests en vuestro sistema!!!
- Obtener la mejor optimizacion de la CPU
  - Usar optimización del compilador
  - Reemplazar float con double
- La idea es ver la mejor configuración para la CPU usada

Cómo optimizar el código paralelo OpenCL

Estrategias Speedup	Plataforma 1		Plataforma 2	
	CPU	GPU	CPU	GPU
Secuencial (not opencl)	1	N/A	1	N/A
C(i,j) por work item, solo memoria global	4,5	3	4,4	4,2
C row per work-item, all global		1,5	3,8	4,7
C row per work-item, A row private		5,2	3,8	9,7
C row per work-item, A row private B row local		15	11,3	9,2

- Seguro se pueden seguir optimizando aspecto en la GPU y en la CPU...!!

Cómo optimizar el código paralelo OpenCL

Estrategias Speedup	Plataforma 1		Plataforma 2	
	CPU	GPU	CPU	GPU
Secuencial (not opencl)	1	N/A	1	N/A
C(i,j) por work item, solo memoria global	4,5	3	4,4	4,2
C row per work-item, all global		1,5	3,8	4,7
C row per work-item, A row private		5,2	3,8	9,7
C row per work-item, A row private B row local		15	11,3	9,2

- Seguro se pueden seguir optimizando aspecto en la GPU y en la CPU...!!
- Qué más?

Resumen: sincronización de work items

- Dentro de un work group void barrier()

Resumen: sincronización de work items

- Dentro de un work group void barrier()
  - Flags CLK_LOCAL_MEM_FENCE y/o CLK_GLOBAL_MEM_FENCE
  - Un work-item que encuentra una barrier() esperará hasta que todos los work-items dentro del grupo alcancen la barrier()
- Entre work groups:
  - No hay garantía qué un grupo terminará antes que otro
  - No hay sincronización como barriers() entre grupos
- Hints para lograr alto desempeño: desde página 93–101 Tutorial

Some examples of *td-problems*

- Multiplicar matrices dividiendo las matrices originales en bloques. Tutorial: 101—118
- Calcular la traspuesta de una matriz A

Some examples of *td-problems*

- Multiplicar matrices dividiendo las matrices originales en bloques. Tutorial: 101—118
- Calcular la traspuesta de una matriz A
- ...

End

Questions?