Software Concurrent
Pràctica de OpenCL

Multiplicació de Matrius

Oscar Amoros Huguet
Universitat de Barcelona 2015

Definició de la pràctica

La multiplicació de matrius es una operació matemàtica bàsica, que conté força paral·lelisme de dades, però també certs problemes de localitat de dades. En aquesta pràctica se os demana que implementeu el codi en OpenCL per a la multiplicació de dues matrius A i B, i retorneu una matriu C amb el resultat de la multiplicació.

Per a simplificar, A i B tindran la mateixa mida per a les dimensions "x" i "y". Junt amb aquest enunciat, teniu una plantilla de codi on trobareu la solució en C sense paral·lelitzar, per comprovar que els resultats en OpenCL són correctes. Podeu utilitzar aquest codi per començar, tot i que haureu de fer forces canvis.

El codi de Host el podeu implementar amb SimpleOpenCL. El codi de Device, és a dir, el Kernel, l'heu de fer amb OpenCL C.

Important: Els resultats de CPU i GPU han de ser els mateixos dins del rang de precisió indicat.

Planificació

Els passos a seguir recomanats son els següents (cada pas, a partir del 1, suma punts):

0) Proveu de compilar i executar els diferents codis d'exemple, i intenteu entendre el que fan.

1) Implementeu un codi que accedeixi al hardware. Mireu com podeu imprimir per pantalla de quin hardware es tracta. Feu un kernel que llegeixi alguna variable d'entrada i escrigui el contingut en una variable de sortida. Un cop executat el codi del kernel, imprimiu per pantalla el resultat escrit a la variable de sortida.

2) Creeu un kernel d'OpenCL, que utilitzant un sol Work Item, implementi el mateix algorisme que el codi que teniu en C estàndard.

3) Modifiqueu el que calgui, per a que aquest kernel utilitzi, com a mínim, tants Work Items com elements hi hagi a l'eix de les "x". Teniu alguns exemples de codi a la carpeta descarregable a la web de SimpleOpenCL. Per a aquesta primera implementació, no feu servir variables de tipus __local, només __global, per reduir complexitat i problemes. Executeu el mateix kernel d'OpenCL tant a la CPU com a la GPU, i compareu el rendiment.

4) Un cop els resultats siguin correctes, intenteu accelerar l'execució, ajustant el numero de Work Items i fent servir variables de tipus __local, que redueixin el nombre d'accessos a la memòria __global. Executeu el codi tant a la CPU com a la GPU i compareu els resultats d'aquesta versió amb l'anterior.

Documentació

Junt amb el codi, haureu d'entregar una documentació que serà el 30% de la nota d'aquesta pràctica. Aquesta documentació haurà d'incloure:

1) Una descripció del problema a solucionar en aquesta pràctica, explicat a la vostra manera.
2) Taules i gràfics mostrant els resultats de temps d'execució de cada versió de codi i cada dispositiu (CPU o GPU). Per a obtenir aquest temps, feu servir la comanda "time" (exemple: time ./programa). El temps que os interessa es el que esta etiquetat com a "real".
3) Conclusions explicant els resultats, i sobretot intentant justificar el perquè dels resultats (no importa que la resposta sigui 100% correcte, importa que estigui raonat en funció de la informació que teniu).

Format de l'entrega

Heu d'entregar totes les versions de codi esmentades a la documentació en una mateixa carpeta, indicant en un fitxer TXT com compilar y quins arxius implementen les versions esmentades als punts 3 i 4. Podeu afegir proves que potser no s'hagin comentat a l'enunciat, però que os hagin servit per a avançar. Es recomana afegir-les, si os han servit per a entendre alguna cosa, ja que també es valorarà la feina feta, i els conceptes apresos.

Aquesta carpeta, i la documentació en PDF, entregueu-les en un mateix ZIP al campus virtual de l'assignatura, abans de la data d'entrega.

Multiplicació de matrius a la GPU

La multiplicació de dues matrius A i B (Figura 1), implica la lectura de posicions de memòria contigües a A, i de posicions allunyades entre si a B. Per a la GPU, és molt important que les lectures a memòria Global, es facin de manera contigua (coalescent).

Una lectura coalescent a la GPU, vol dir que cada Work Item, estarà llegint una posició de memòria corresponent al seu índex. Per exemple:

WI 1 -> llegeix el valor A[33]
WI 2 -> llegeix el valor A[34]
WI 3 -> llegeix el valor A[35]
WI 4 -> llegeix el valor A[36]

Aquesta seria una lectura a memòria, coalescent. El següent exemple no ho seria:

WI 1 -> llegeix el valor A[12]
WI 2 -> llegeix el valor A[3]
WI 3 -> llegeix el valor A[24]
WI 4 -> llegeix el valor A[11]

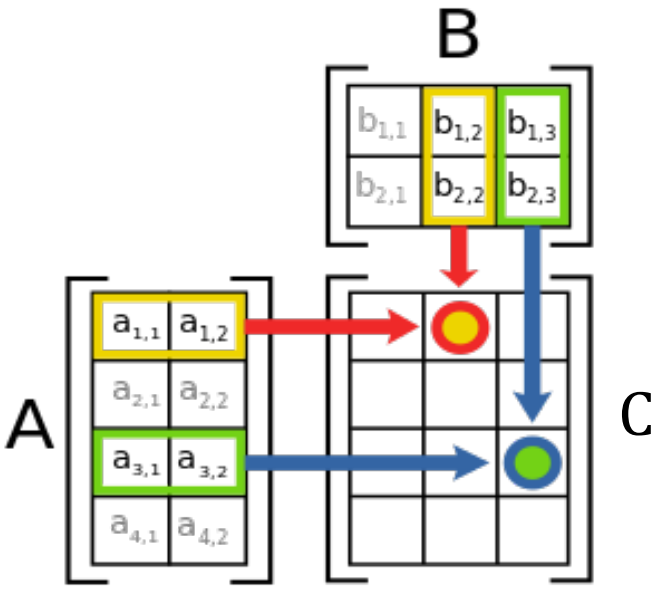

Figura 1: Multiplicació de matrius.

Com es veu a la Figura 1, podem assignar un Work Item per cada element de la matriu C. Cada un d'aquests Work Items, s'encarregarà de fer la multiplicació d'una fila de A per una columna de B, obtenint un sol valor com a resultat.

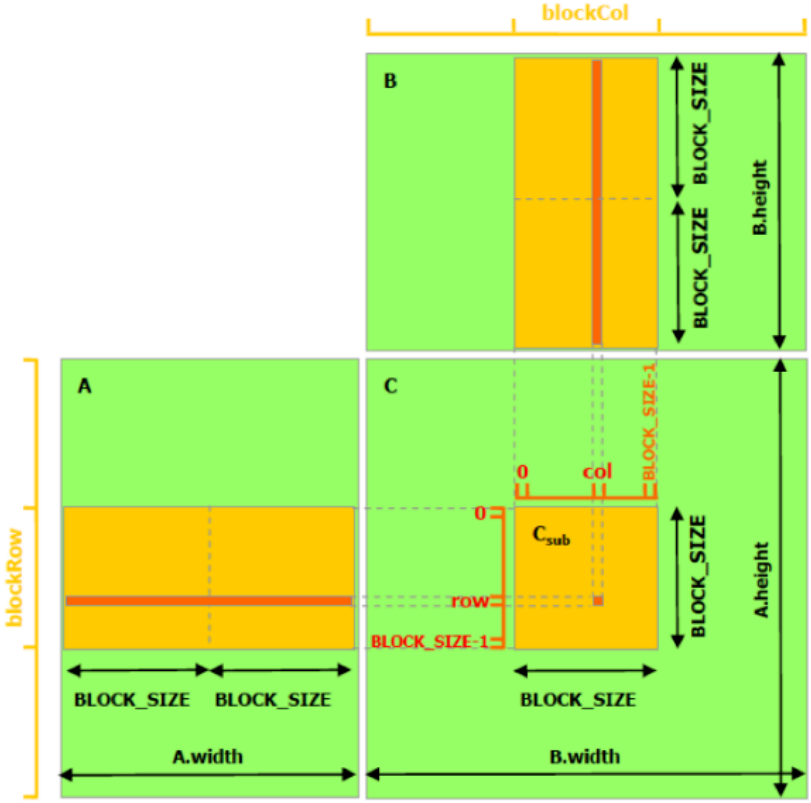

Figura 2: Distribució de Work Items i Work Groups (Blocks)

La figura 2, suggereix subdividir l'espai de Work Items en Work Groups (Blocks). Les dimensions dels Work Groups ideals varien entre GPU's, però en tot cas han de

ser múltiples de les dimensions de la matriu C, o fer coses al codi que no os demanarem en aquesta pràctica.

A les diapositives de classe, trobareu detalls sobre els Work Items i els Work Groups.

Data d'entrega

La data límit d'entrega és el diumenge 29 de Març.