CentraleSupélec

Mineure HPC-SBD

Bibliothèque CUBLAS

Stéphane Vialle

Stephane.Vialle@centralesupelec.fr
http://www.metz.supelec.fr/~vialle

CentraleSupélec

Bibliothèques BLAS et CUBLAS

1 – Les bibliothèques BLAS
2 – CUBLAS *vs* BLAS
3 – La fonction *cublasDgemm*
4 – Format des données de *cublasDgemm*
5 – Espace de stockage pour *cublasDgemm*

Les bibliothèques « BLAS »

BLAS : Basic Linear Algebra Subprograms

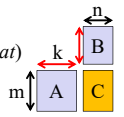
- Ensemble de fonctions d'algèbre linéaire
- Prototypes publiés en 1979
  https://fr.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms
- 3 niveaux de BLAS :
  - niveau 1 : opérations sur les vecteurs
  - niveau 2 : opérations de type matrice – vecteur
  - niveau 3 : opérations de type matrice – matrice
- **UNE API standardisée et DES implantations**
  - implantations open-sources
    ex : OpenBLAS, ATLAS (très efficaces)
  - implantations propriétaires ciblées sur un type de matériel
    ex : bibliothèque MKL d'Intel
  - installer en recompilant sur la machine cible

Les bibliothèques « BLAS »

La Fonction `cblas_dgemm` :

- Fonction de produit de matrices denses
- Travaille sur des *double* (`cblas_sgemm` pour les *float*)

$$C = \alpha.op(A) \times op(B) + \beta.C$$




```
void cblas_dgemm (
    const CBLAS_LAYOUT Layout,
    const CBLAS_TRANSPOSE transa,
    const CBLAS_TRANSPOSE transb,
    const int m, const int n, const int k,
    const double alpha,
    const double *a, const int lda,
    const double *b, const int ldb,
    const double beta,
    double *c, const int ldc)
```

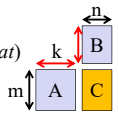
Stockage en *row major* ou *column major*

Les bibliothèques « BLAS »

La Fonction `cblas_dgemm` :

- Fonction de produit de matrices denses
- Travaille sur des *double* (`cblas_sgemm` pour les *float*)

$$C = \alpha.op(A) \times op(B) + \beta.C$$




```
void cblas_dgemm (
    const CBLAS_LAYOUT Layout,
    const CBLAS_TRANSPOSE transa,
    const CBLAS_TRANSPOSE transb,
    const int m, const int n, const int k,
    const double alpha,
    const double *a, const int lda,
    const double *b, const int ldb,
    const double beta,
    double *c, const int ldc)
```

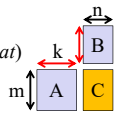
Transposition « au vol » (ou pas) de A et B

Les bibliothèques « BLAS »

La Fonction `cblas_dgemm` :

- Fonction de produit de matrices denses
- Travaille sur des *double* (`cblas_sgemm` pour les *float*)

$$C = \alpha.op(A) \times op(B) + \beta.C$$




```
void cblas_dgemm (
    const CBLAS_LAYOUT Layout,
    const CBLAS_TRANSPOSE transa,
    const CBLAS_TRANSPOSE transb,
    const int m, const int n, const int k,
    const double alpha,
    const double *a, const int lda,
    const double *b, const int ldb,
    const double beta,
    double *c, const int ldc)
```

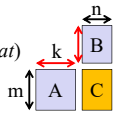
Tailles des matrices

Les bibliothèques « BLAS »

La Fonction `cblas_dgemm` :

- Fonction de produit de matrices denses
- Travaille sur des *double* (`cblas_sgemm` pour les *float*)

$$C = \alpha.op(A) \times op(B) + \beta.C$$




```
void cblas_dgemm (
    const CBLAS_LAYOUT Layout,
    const CBLAS_TRANSPOSE transa,
    const CBLAS_TRANSPOSE transb,
    const int m, const int n, const int k,
    const double alpha,
    const double *a, const int lda,
    const double *b, const int ldb,
    const double beta,
    double *c, const int ldc)
```

lda, ldb, ldc : tailles des zones de stockage (voir plus loin)

2 – CUBLAS *vs* BLAS

- Mêmes fonctionnalités d'algèbre linéaire que les BLAS
- Mais quelques différences :

Besoin d'initialiser l'usage de la bibliothèque

- Récupération d'un « *handle* » sur la bibliothèque
- Passage de ce *handle* en paramètre de toutes les fonctions CUBLAS

Le format « *column major* » (style FORTRAN) est imposé

- Possibilité de convertir les données d'entrée au vol
- Besoin de convertir les données de sortie

La bibliothèque ne se charge pas des transferts CPU/GPU

→ les transferts restent à la charge du développeur

Quelques nouvelles fonctions disponibles

→ des « extensions des BLAS » : non standard mais pratiques !

2 – CUBLAS *vs* BLAS

Utilisation des CUBLAS

1. Ajout de `#include <cublas_v2.h>` au début du fichier src Cuda
2. Ajout de `-lcublas` à l'édition de lien
3. Initialisation des CUBLAS dans le code Cuda

```
cublasHandle_t handle;
cublasCreate(&handle); // Return a cudaStatus_t
                       //    CUBLAS_STATUS_SUCCESS ?
cublasXxxx(handle,…);  // Cublas usage
cublasDestroy(handle); // End of cublas usage
```

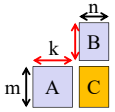
Rmq : L'initialisation (`cublasCreate`) peut prendre un peu de temps! Ne la faire qu'une seule fois au début du pgm.

3 – La fonction *cublasDgemm*

API :

```
cublasStatus_t cublasDgemm(cublasHandle_t handle,
                           cublasOperation_t transa,
                           cublasOperation_t transb,
                           int m, int n, int k,
                           const double *alpha,
                           const double *A, int lda,
                           const double *B, int ldb,
                           const double *beta,
                           double *C, int ldc)
```

$$C = \alpha.op(A) \times op(B) + \beta.C$$



Comparé aux BLAS :

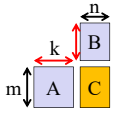
- On passe le « *handle* » des cublas (1er paramètre)
- Les valeurs « *alpha* » et « *beta* » ne sont pas des constantes mais des adresse de variables (constantes)
- On ne précise pas le format des données :
  → le stockage « *column major* » est imposé

3 – La fonction *cublasSgemm*

API :

```
cublasStatus_t cublasDgemm(cublasHandle_t handle,
                           cublasOperation_t transa,
                           cublasOperation_t transb,
                           int m, int n, int k,
                           const float *alpha,
                           const float *A, int lda,
                           const float *B, int ldb,
                           const float *beta,
                           float *C, int ldc)
```

$$C = \alpha.op(A) \times op(B) + \beta.C$$



Comparé aux BLAS :

- On passe le « *handle* » des cublas (1er paramètre)
- Les valeurs « *alpha* » et « *beta* » ne sont pas des constantes mais des adresse de variables (constantes)
- On ne précise pas le format des données :
  → le stockage « *column major* » est imposé

4 – Format des données de *cublasDgemm*

Format des données (matrices) :

$$C = \alpha.op(A) \times op(B) + \beta.C$$

```
cublasDgemm(
  cublasHandle_t handle,
  cublasOperation_t transa,
  cublasOperation_t transb,
  int m, int n, int k,……)
```

A est une matrice :

- mathématiquement de *m lignes* × *k colonnes*,
- considérée stockée au format *column major* (imposé),
- que l'on peut transposer « au vol » :

$$op(A) = \begin{cases} A & \text{if transa == CUBLAS\_OP\_N} \\ A^T & \text{if transa == CUBLAS\_OP\_T} \\ A^H & \text{if transa == CUBLAS\_OP\_C} \end{cases}$$ → matrice conjuguée

Idem pour B : matrice de *k* lignes × *n colonnes*

4 – Format des données de *cublasDgemm*

Format des données (matrices) :

$C = \alpha.op(A) \times op(B) + \beta.C$

```
cublasDgemm(
  cublasHandle_t handle,
  cublasOperation_t transa,
  cublasOperation_t transb,
  int m, int n, int k,.....)
```

A est une matrice :

- mathématiquement de *m lignes* × *k colonnes*,
- considérée stockée au format *column major* (imposé),
- que l'on peut transposer « au vol » :

En langage C/C++ (comme en CUDA) un tableau 2D est stocké en *row major*…

…en demandant la transposition au vol d'une matrice (avec `transa = CUBLAS_OP_T`) on l'obtient en *column major* !

Bibliothèques BLAS et CUBLAS

4 – Format des données de *cublasDgemm*

Format des données (matrices) :

$C = \alpha.op(A) \times op(B) + \beta.C$

```
cublasDgemm(
  cublasHandle_t handle,
  cublasOperation_t transa,
  cublasOperation_t transb,
  int m, int n, int k,.....)
```

Mais la matrice résultat (C) :

- est générée au format *column major* (imposé)
- donc on obtient C^T au format naturel du langage C/C++/CUDA

Pour obtenir C au format du C/C++/CUDA on peut :

- Transposer C avec un kernel écrit pour l'occasion
- Transposer C avec la fonction CUBLAS `cudaDgeam` qui est une extension des BLAS
- Organiser les calculs matriciels pour calculer mathématiquement C^T et donc obtenir C dans le format du C/C++/CUDA

Bibliothèques BLAS et CUBLAS

4 – Format des données de *cublasDgemm*

Kernel de transposition simpliste :

```
__global__ void TransposeKernel_v0(float *MT, float *M,
                                   int mLig, int nCol)
{
  int lig = threadIdx.y + blockIdx.y*BSIZE_XY_KT0;
  int col = threadIdx.x + blockIdx.x*BSIZE_XY_KT0;

  if (lig < mLig && col < nCol)
    MT[col*mLig + lig] = M[lig*nCol + col];
}
```

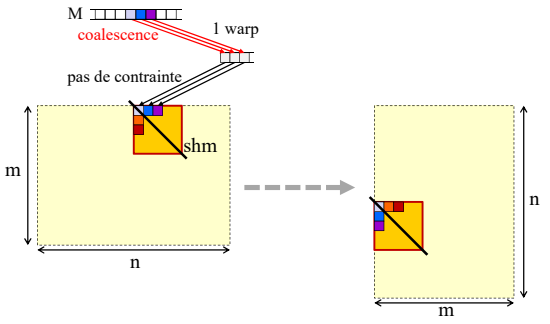
Accès NON coalescent — Accès coalescent

Solution : passer par la shared memory pour être coalescent à la lecture ET à l'écriture…

Bibliothèques BLAS et CUBLAS

4 – Format des données de *cublasDgemm*

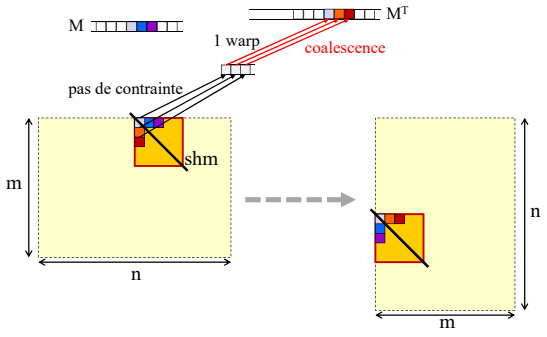
Kernel de transposition coalescent :



Bibliothèques BLAS et CUBLAS

4 – Format des données de *cublasDgemm*

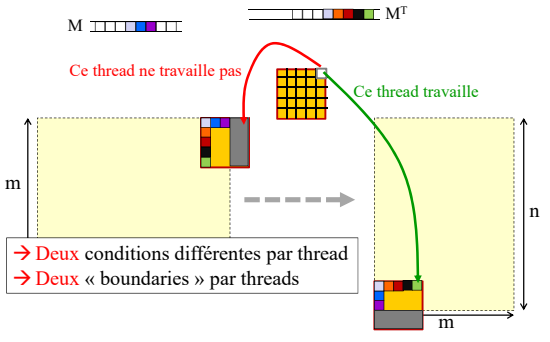
Kernel de transposition coalescent :



Bibliothèques BLAS et CUBLAS

4 – Format des données de *cublasDgemm*

Kernel de transposition coalescent :



4 – Format des données de *cublasDgemm*

Kernel de transposition coalescent :

```
__global__ void TransposeKernel_v1(float *MT, float *M,
                                   int mLig, int nCol)
{
int firstLibBlock = blockIdx.y*BSIZE_XY_KT1;
int firstColBlock = blockIdx.x*BSIZE_XY_KT1;
int lig = firstLibBlock + threadIdx.y;
int col = firstColBlock + threadIdx.x;
int ligT = firstColBlock + threadIdx.y;
int colT = firstLibBlock + threadIdx.x;
__shared__ float shM[BSIZE_XY_KT1][BSIZE_XY_KT1];
if (lig < mLig && col < nCol)    // Condition 1
  shM[threadIdx.y][threadIdx.x] = M[lig*nCol + col];
__syncthreads();
if (ligT < nCol && colT < mLig) // Condition 2
  MT[ligT*mLig + colT] = shM[threadIdx.x][threadIdx.y];
}
```


4 – Format des données de *cublasDgemm*

Transposition par `cublasDgeam` :

Réalise le calcul de : C = α.op(A) + β.op(B)

```
cublasStatus_t cublasDgeam(
      cublasHandle_t handle,
      cublasOperation_t transa,
      cublasOperation_t transb,
      int m, int n,
      const double *alpha,
      const double *A, int lda,
      const double *beta,
      const double *B, int ldb,
      double *C, int ldc)
```

Voir aussi `cublasSgeam(…)`

→ Permet le calcul (très rapide) d'une transposée de matrice

Rmq : on peut passer un ptr `NULL` si on ne se sert pas d'une matrice et qu'on ne fait pas d'opération dessus

→ TP 4

4 – Format des données de *cublasDgemm*

Calcul de M^T en stockage *column major* :

Calculer M^T au lieu de M avec les CUBLAS

→ Donc obtenir M^T stockée en *column major*

→ Donc obtenir M dans un tableau C/C++/CUDA en *row major*

Rappel : $(A \times B)^T = B^T \times A^T$

.....

→ TP 4

5 – Espace de stockage pour *cublasDgemm*

Autres paramètres (doc NVIDIA)

Les tableaux de stockages de A, B et C, peuvent être plus grands que les matrices

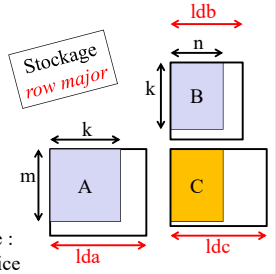
On doit préciser leur dimension principale…

Param.	Memory	In/out	Meaning
handle		input	handle to the CUBLAS library context.
transa		input	operation op(A) that is non- or (conj.) transpose.
transb		input	operation op(B) that is non- or (conj.) transpose.
m		input	number of rows of matrix op(A) and C.
n		input	number of columns of matrix op(B) and C.
k		input	number of columns of op(A) and rows of op(B).
alpha	host or device	input	<type> scalar used for multiplication.
A	device	input	<type> array of dimensions `lda x k` with `lda>=max(1,m)` if `transa == CUBLAS_OP_N` and `lda x m` with `lda>=max(1,k)` otherwise.
lda		input	leading dimension of two-dimensional array used to store the matrix A.
B	device	input	<type> array of dimension `ldb x n` with `ldb>=max(1,k)` if `transa == CUBLAS_OP_N` and `ldb x k` with `ldb>=max(1,n)` otherwise.
ldb		input	leading dimension of two-dimensional array used to store matrix B.
beta	host or device	input	<type> scalar used for multiplication. If `beta==0`, C does not have to be a valid input.
C	device	in/out	<type> array of dimensions `ldc x n` with `ldc>=max(1,m)`.
ldc		input	leading dimension of a two-dimensional array used to store the matrix C.

5 – Espace de stockage pour *cublasDgemm*

Les espaces de stockage peuvent être plus grand que les matrices :

- Réutilisation d'un espace alloué précédemment
- Espace de stockage dimensionné au maximum
- Calcul sur des sous-matrices
- …



La fonction `cblas_dgemm` doit connaitre :

- le nombre de colonnes de la matrice
- le nombre de colonnes de l'espace de stockage

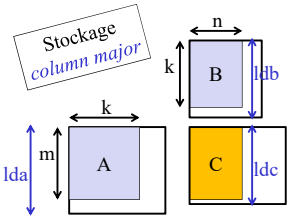
→ **Pour savoir combien d'espace sauter entre deux lignes**

Ex : en *row major* sans transposer A et B :
il faut sauter lda – k, avec : lda ≥ k, ldb ≥ n, ldc ≥ n

5 – Espace de stockage pour *cublasDgemm*

Les espaces de stockage peuvent être plus grand que les matrices :

- Réutilisation d'un espace alloué précédemment
- Espace de stockage dimensionné au maximum
- Calcul sur des sous-matrices
- …



La fonction `cublasDgemm` impose un stockage en *column major* :

- Sans transposer A et B il faut : lda ≥ m, ldb ≥ k, ldc ≥ m
- En transposant A il faut : lda ≥ k
- En transposant B il faut : ldb ≥ n

5 – Espace de stockage pour *cublasDgemm*

Spécification de `lda` et `ldb`

A	device	input	<type> array of dimensions `lda x k` with `lda>=max(1,m)` if `transa == CUBLAS_OP_N` and `lda x m` with `lda>=max(1,k)` otherwise.
lda		input	leading dimension of two-dimensional array used to store the matrix `A`.
B	device	input	<type> array of dimension `ldb x n` with `ldb>=max(1,k)` if `transa == CUBLAS_OP_N` and `ldb x k` with `ldb>=max(1,n)` otherwise.
ldb		input	leading dimension of two-dimensional array used to store matrix `B`.

- Un tableau A d'un espace total de $m \times k$ eléments suffira
- Si on n'a pas transposé A on indiquera `lda` = m (ou plus), Sinon on indiquera `lda` = k (ou plus)
- Un tableau B d'un espace total de $k \times n$ eléments suffira
- Si on n'a pas transposé B on indiquera `ldb` = k (ou plus), Sinon on indiquera `lda` = n (ou plus)

5 – Espace de stockage pour *cublasDgemm*

Spécification de `ldc`

C	device	in/out	<type> array of dimensions `ldc x n` with `ldc>=max(1,m)`.
ldc		input	leading dimension of a two-dimensional array used to store the matrix `c`.

- Un tableau C d'un espace total de $m \times n$ double suffira
- C sera (forcément) en *column major*
- On indiquera `ldc` = m (ou plus)

Bibliothèques BLAS et CUBLAS

FIN