Devoir No. 6 et Lab. # 6

PRÉ-REQUIS

- Vous devez avoir complété le Pré-Lab 3 et lu les chapitres 2 à 5 de la référence [2]
- À faire avant de commencer le lab (Voir annexe A)
  1. Ajouter “cl.exe” aux variables d’environnement du système

PARTIE 1 - MULTIPLICATION MATRICE-MATRICE

Objectif : Écrire une multiplication matricielle C = A x B. Par souci de simplicité, nous allons uniquement utiliser des matrices carrées dont les éléments sont des single-precision floating- point.

Méthodologie :

Suivez les instructions ci-dessous :

A. Écrire une fonction “host-stub ”qui allouera la mémoire pour les matrices d’entrées et la matrice de sortie, fera le transfert de ces matrices vers le Device, lancera le kernel, fera le transfert de la matrice de sortie vers l’Host et libèrera la mémoire des matrices d’entrées et de sortie sur le Device. Laissez les configurations d’exécution vide pour cette étape.

Cette fonction a quelque peu changé depuis mon laboratoire #5 :

```
// Helper function for using CUDA to add vectors in parallel.
cudaError_t multiplyWithCuda(float* c, const float* a, const float* b, unsigned int size, kernelType type)
{
    float* dev_a = 0;
    float* dev_b = 0;
    float* dev_c = 0;
    cudaError_t cudaStatus;

    // Choose which GPU to run on, change this on a multi-GPU system.
    cudaStatus = cudaSetDevice(0);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaSetDevice failed!  Do you have a CUDA-capable GPU installed?");
        goto Error;
    }

    // Allocate GPU buffers for three vectors (two input, one output)    .
    cudaStatus = cudaMalloc((void**)&dev_c, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }

    cudaStatus = cudaMalloc((void**)&dev_a, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }

    cudaStatus = cudaMalloc((void**)&dev_b, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }

    // Copy input vectors from host memory to GPU buffers.
    cudaStatus = cudaMemcpy(dev_a, a, (size * size) * sizeof(float), cudaMemcpyHostToDevice);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed!");
        goto Error;
    }


    cudaStatus = cudaMemcpy(dev_b, b, (size * size) * sizeof(float), cudaMemcpyHostToDevice);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed!");
        goto Error;
    }
```

```
// Launch a kernel on the GPU
if (size == 4) {
    dim3 threadsParBlock(size, size, 1);
    dim3 nombreDeBlock(1, 1, 1);
    switch (type) {
        case KERNEL_ELEMENT:
            multiplyKernelElement << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;
        case KERNEL_ROW:
            multiplyKernelRow << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;
        case KERNEL_COLUMN:
            multiplyKernelColumn << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;

        default:
            break;
    }


}

else if (size >= minMatrixWidth && size <= maxMatrixWidth) {
    dim3 threadsParBlock(size, size, 1);
    dim3 nombreDeBlock((size / blockWidth), (size / blockWidth), 1);

    switch (type) {
        case KERNEL_ELEMENT:
            multiplyKernelElement << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;
        case KERNEL_ROW:
            multiplyKernelRow << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;
        case KERNEL_COLUMN:
            multiplyKernelColumn << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);
            break;

        default:
            break;
    }
}


else {
    printf("Undefined behaviour. Please use matrix size between 8 and 64\n");
    return cudaErrorInvalidValue;
}

  else {
      printf("Undefined behaviour. Please use matrix size between 8 and 64\n");
      return cudaErrorInvalidValue;
  }

  // Check for any errors launching the kernel
  cudaStatus = cudaGetLastError();
  if (cudaStatus != cudaSuccess) {
      fprintf(stderr, "addKernel launch failed: %s\n", cudaGetErrorString(cudaStatus));
      goto Error;
  }

  // cudaDeviceSynchronize waits for the kernel to finish, and returns
  // any errors encountered during the launch.
  cudaStatus = cudaDeviceSynchronize();
  if (cudaStatus != cudaSuccess) {
      fprintf(stderr, "cudaDeviceSynchronize returned error code %d after launching addKernel!\n", cudaStatus);
      goto Error;
  }

  // Copy output vector from GPU buffer to host memory.
  cudaStatus = cudaMemcpy(c, dev_c, (size * size) * sizeof(float), cudaMemcpyDeviceToHost);
  if (cudaStatus != cudaSuccess) {
      fprintf(stderr, "cudaMemcpy failed!");
      goto Error;
  }

Error:
    cudaFree(dev_c);
    cudaFree(dev_a);
    cudaFree(dev_b);

    return cudaStatus;
}
```

Comme vous pouvez constater, dépendamment de la grosseur de la matrice entrée, le kernell sera calculé différemment. Dans ce programme, je peux faire la multiplication matricielle sur des matrices de 4*4, et tout autre matrice carrée qui ont une dimension entre 8 et 64. Si une taille plus grande est rentrée, un message est écrit pour indiquer que ce calcul de matrice n'est pas pris en compte dans ce programme.

Pour rendre ce code plus agréable à lire, j'ai créé un enum au début de mon programme pour indiquer les différents types de calcul il est possible de faire avec les threads.

```
typedef enum {
    KERNEL_ELEMENT = 0,
    KERNEL_ROW,
    KERNEL_COLUMN
} kernelType;
```

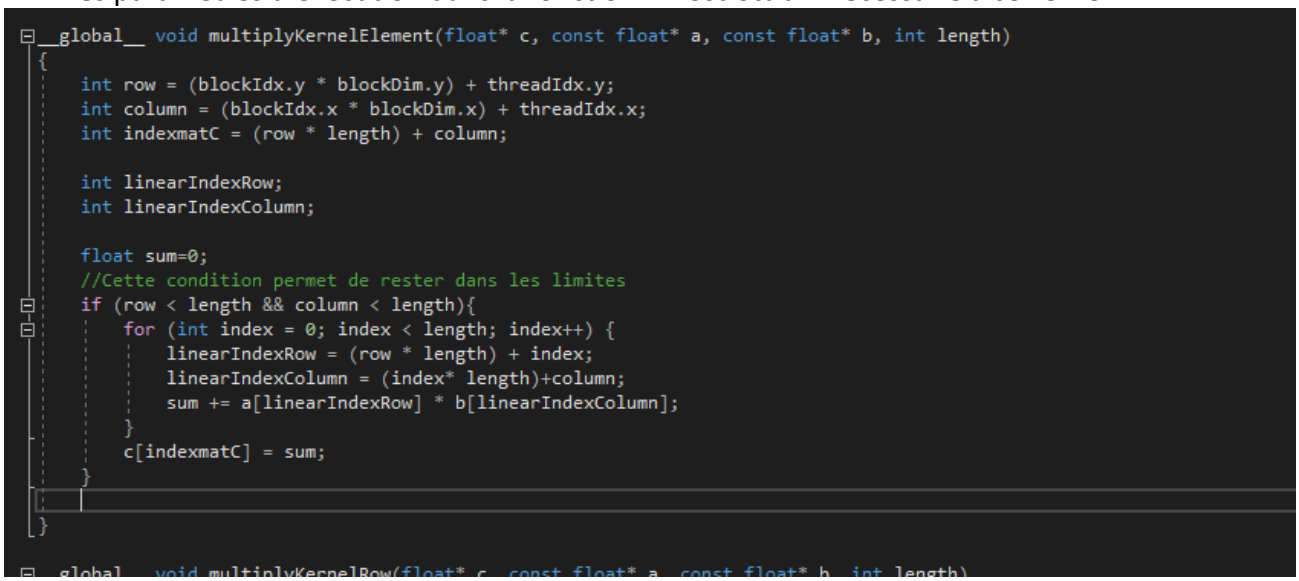
Lorsqu'on appelle la fonction multiplyWithCuda, il est donc important de spécifier quel type d'opération qu'on veut faire tel qu'énoncé dans le cahier dans le laboratoire.

J'ai aussi créé des variables constantes nous permettant de rapidement changer différentes fonctionnalités du programme :

```
const int blockWidth = 8;
const int minMatrixWidth = 8;
const int maxMatrixWidth = 64;
```

J'ai aussi corrigé l'allocation de la mémoire pour correspondre au float.

B. **Écrivez un kernel où chaque thread calculera un seul élément de la matrice de sortie.** Ajouter les paramètres d'exécution dans la fonction « Host-Stub » nécessaire à ce kernel.



```
__global__ void multiplyKernelElement(float* c, const float* a, const float* b, int length)
{
    int row = (blockIdx.y * blockDim.y) + threadIdx.y;
    int column = (blockIdx.x * blockDim.x) + threadIdx.x;
    int indexmatC = (row * length) + column;

    int linearIndexRow;
    int linearIndexColumn;

    float sum=0;
    //Cette condition permet de rester dans les limites
    if (row < length && column < length){
        for (int index = 0; index < length; index++) {
            linearIndexRow = (row * length) + index;
            linearIndexColumn = (index* length)+column;
            sum += a[linearIndexRow] * b[linearIndexColumn];
        }
        c[indexmatC] = sum;
    }

}

__global__ void multiplyKernelRow(float* c, const float* a, const float* b, int length)
```

Cette fonction permet de calculer un élément de la matrice de sortie. Contrairement au devoir 5, chaque thread doit quand même faire une loop car la matrice de sortie est une somme de toutes les multiplications rangées par colonnes de chaque élément des matrices A et B.

C. **Écrivez un kernel où chaque thread calculera tous les éléments d'une ligne de la matrice de sortie.** Ajouter les paramètres d'exécution dans la fonction « Host-Stub » nécessaire à ce kernel.

```
__global__ void multiplyKernelRow(float* c, const float* a, const float* b, int length)
{
    int row = (blockIdx.y * blockDim.y) + threadIdx.y;
    int indexMatC = (row * length);

    int linearIndexRow;
    int linearIndexColumn;

    //Cette condition permet de rester dans les limites
    if (row < length) {

        for (int column = 0; column < length; column++) {
            float sum = 0;
            for (int index = 0; index < length; index++) {
                linearIndexRow = (row * length) + index;
                linearIndexColumn = (index * length) + column;
                sum += a[linearIndexRow] * b[linearIndexColumn];
            }
            c[(indexMatC + column)] = sum;
        }


    }

}
```

Pour faire ceci, il est inévitable de faire une loop imbriqué dans une autre loop, car on doit calculer tous les élélements de la matrice de sortie. En tant que tel, le calcul reste très similaire que la question précédente dans la loop imbriqué, sauf que la première loop va garder la colonne respectif de la matrice de sortie, car on doit calculer une rangée complète dans ce thread.

D. **Écrivez un kernel où chaque thread calculera tous les éléments d'une colonne de la matrice de sortie.** Ajouter les paramètres d'exécution dans la fonction « Host-Stub » nécessaire à ce kernel.

```
__global__ void multiplyKernelColumn(float* c, const float* a, const float* b, int length)
{
    int column = (blockIdx.x * blockDim.x) + threadIdx.x;
    int linearIndexRow;
    int linearIndexColumn;

    //Cette condition permet de rester dans les limites
    if (column < length) {

        for (int row = 0; row < length; row++) {
            float sum = 0;
            for (int index = 0; index < length; index++) {
                linearIndexRow = (row * length) + index;
                linearIndexColumn = (index * length) + column;
                sum += a[linearIndexRow] * b[linearIndexColumn];
            }
            c[(row*length)+column] = sum;
        }


    }
}
```

Cette fois-ci, l'implémentation est très similaire que la question précédente, sauf qu'il faut calculer une colonne pour la matrice de sortie. Pour faire ceci, mathématiquement il faut calculer chaque rangée de la matrice A pour une colonne de la matrice B, et ceci donnera une colonne de la matrice C.

E. Analysez les avantages et inconvénients de chaque kernel ci-dessus.

Comme il est possible de voir dans les captures d'écran plus bas, faire un calcul élément par élément est énormément plus rapide que le calcul colonne ou rangée. Autre que cela, le calcul par colonne versus par rangée ne présente un avantage si significatif en terme de rapidité d'éxecution dans ce contexte

Pour A à E,

A. Validez que le kernel est fonctionnel et effectue le bon calcul en utilisant un simple programme en CUDA C où :
    a. **Les matrices doivent avoir une taille variable 4x4, 8x8, 32x32 et 64x64.**

Matrice 4*4

Élément :

```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_ELEMENT);
```

```
Matrix A:
{ 0.01 2.25 0.77 3.23 }
{ 2.34 1.92 1.40 3.58 }
{ 3.29 2.99 0.70 3.44 }
{ 2.84 2.05 1.22 0.06 }

Matrix B:
{ 0.37 1.46 0.59 0.66 }
{ 3.95 1.78 0.48 0.02 }
{ 0.04 1.51 2.13 2.28 }
{ 2.41 2.43 0.66 2.65 }

Matrix C:
{ 16.73 13.05 4.87 10.39 }
{ 17.12 17.66 7.66 14.30 }
{ 21.31 19.52 7.13 12.94 }
{ 9.35 9.79 5.28 4.86 }
```

Rangée :

```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_ROW);
```

```
Matrix A:
{ 0.01 2.25 0.77 3.23 }
{ 2.34 1.92 1.40 3.58 }
{ 3.29 2.99 0.70 3.44 }
{ 2.84 2.05 1.22 0.06 }

Matrix B:
{ 0.37 1.46 0.59 0.66 }
{ 3.95 1.78 0.48 0.02 }
{ 0.04 1.51 2.13 2.28 }
{ 2.41 2.43 0.66 2.65 }

Matrix C:
{ 16.73 13.05 4.87 10.39 }
{ 17.12 17.66 7.66 14.30 }
{ 21.31 19.52 7.13 12.94 }
{ 9.35 9.79 5.28 4.86 }
```

Colonne :

```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_COLUMN);
```

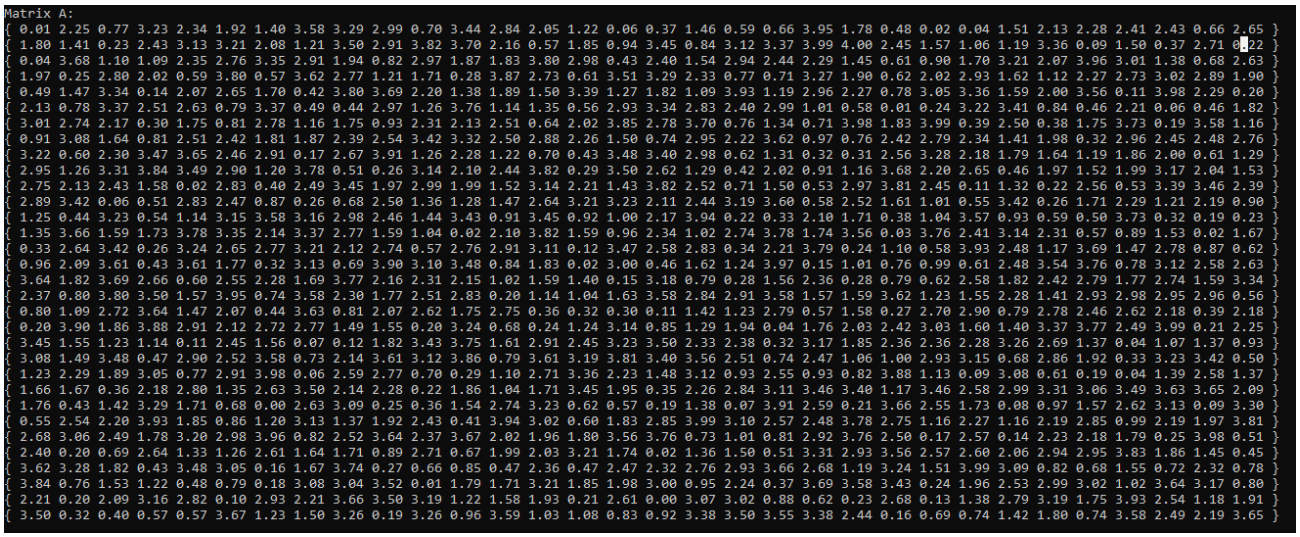
```
Matrix A:
{ 0.01 2.25 0.77 3.23 }
{ 2.34 1.92 1.40 3.58 }
{ 3.29 2.99 0.70 3.44 }
{ 2.84 2.05 1.22 0.06 }

Matrix B:
{ 0.37 1.46 0.59 0.66 }
{ 3.95 1.78 0.48 0.02 }
{ 0.04 1.51 2.13 2.28 }
{ 2.41 2.43 0.66 2.65 }

Matrix C:
{ 16.73 13.05 4.87 10.39 }
{ 17.12 17.66 7.66 14.30 }
{ 21.31 19.52 7.13 12.94 }
{ 9.35 9.79 5.28 4.86 }
```

Matrice 32*32

Élément :

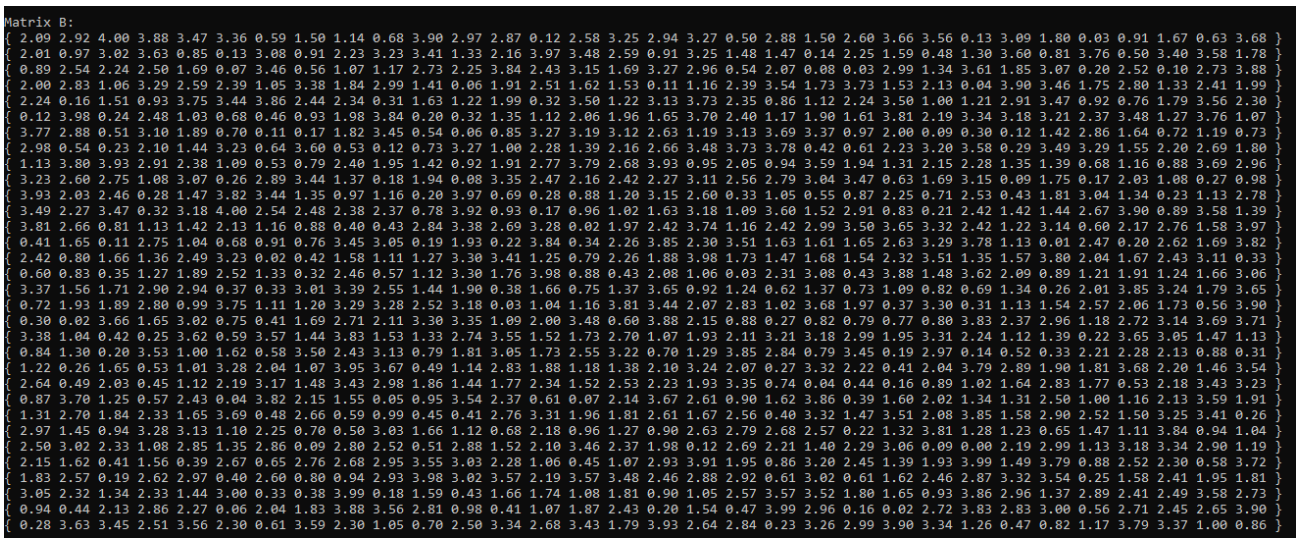
```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_ELEMENT);
```

Matrice A :



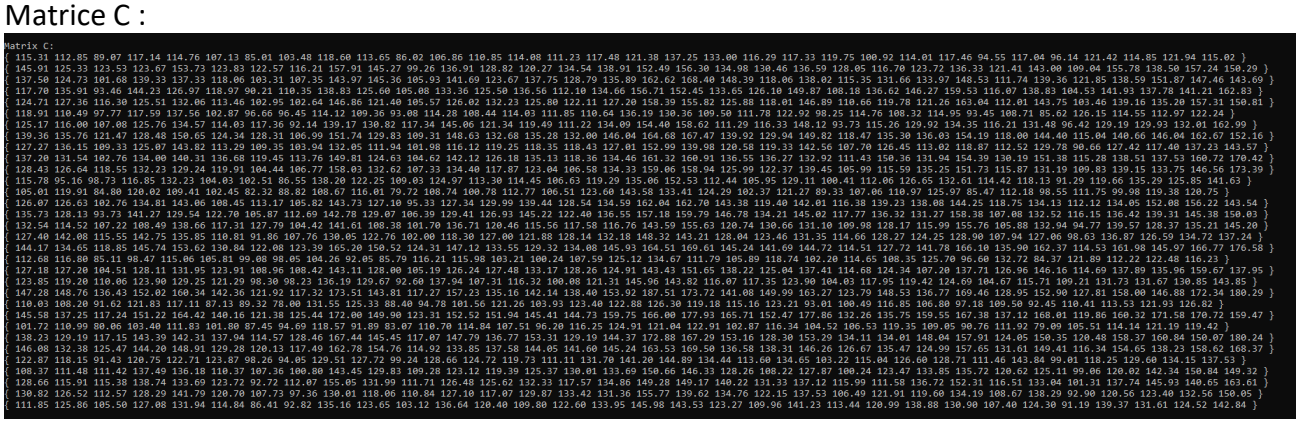
```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }
{ 3.64 1.82 3.69 2.66 0.60 2.55 2.28 1.69 3.77 2.16 2.31 2.15 1.02 1.59 1.40 0.15 3.18 0.79 0.28 1.56 2.36 0.28 0.79 0.62 2.58 1.82 2.42 2.79 1.77 2.74 1.59 3.34 }
{ 2.37 0.80 3.80 3.50 1.57 3.95 0.74 3.58 2.30 1.77 2.51 2.83 0.20 1.14 1.04 1.63 3.58 2.84 2.91 3.58 1.57 1.59 3.62 1.23 1.55 2.28 1.41 2.93 2.98 2.95 2.96 0.56 }
{ 0.80 1.09 2.72 3.64 1.47 2.07 0.44 3.63 0.81 2.07 2.62 1.75 2.75 0.36 0.32 0.30 0.11 1.42 1.23 2.79 0.57 1.58 0.27 2.70 2.90 0.79 2.78 2.46 2.62 2.18 0.39 2.18 }
{ 0.20 3.90 1.86 3.88 2.91 2.12 2.72 2.77 1.49 1.55 0.20 3.24 0.68 0.24 1.24 3.14 0.85 1.29 1.94 0.04 1.76 2.03 2.42 3.03 1.60 1.40 3.37 3.77 2.49 3.99 0.21 2.25 }
{ 3.45 1.55 1.23 1.14 0.11 2.45 1.56 0.07 0.12 1.82 3.43 3.75 1.61 2.91 2.45 3.23 3.50 2.33 2.38 0.32 3.17 1.85 2.36 2.36 2.28 3.26 2.69 1.37 0.04 1.07 1.37 0.93 }
{ 3.08 1.49 3.48 0.47 2.90 2.52 3.58 0.73 2.14 3.61 3.12 3.86 0.79 3.61 3.19 3.81 3.40 3.56 2.51 0.74 2.47 1.06 1.00 2.93 3.15 0.68 2.86 1.92 0.33 3.23 3.42 0.50 }
{ 1.23 2.29 1.89 3.05 0.77 2.91 3.98 0.06 2.59 2.77 0.70 0.29 1.10 2.71 3.36 2.23 1.48 3.12 0.93 2.55 0.93 0.82 3.88 1.13 0.09 3.08 0.61 0.19 0.04 1.39 2.58 1.37 }
{ 1.66 1.67 0.36 2.18 2.80 1.35 2.63 3.50 2.14 2.28 0.22 1.86 1.04 1.71 3.45 1.95 0.35 2.26 2.84 3.11 3.46 3.40 1.17 3.46 2.58 2.99 3.31 3.06 3.49 3.63 3.65 2.09 }
{ 1.76 0.43 1.42 3.29 1.71 0.68 0.00 2.63 3.09 0.25 0.36 1.54 2.74 3.23 0.62 0.57 0.19 1.38 0.07 3.91 2.59 0.21 3.66 2.55 1.73 0.08 0.97 1.57 2.62 3.13 0.09 3.30 }
{ 0.55 2.54 2.20 3.93 1.85 0.86 1.20 3.13 1.37 1.92 2.43 0.41 3.94 3.02 0.60 1.83 2.85 3.99 3.10 2.57 2.48 3.78 2.75 1.16 2.27 1.16 2.19 2.85 0.99 2.19 1.97 3.81 }
{ 2.68 3.06 2.49 1.78 3.20 2.98 3.96 0.82 2.52 3.64 2.37 3.67 2.02 1.96 1.80 3.56 3.76 0.73 1.01 0.81 2.92 3.76 2.50 0.17 2.57 0.14 2.23 2.18 1.79 0.25 3.98 0.51 }
{ 2.40 0.20 0.69 2.64 1.33 1.26 2.61 1.64 1.71 0.89 2.71 0.67 1.99 2.03 3.21 1.74 0.02 1.36 1.50 0.51 3.31 2.93 3.56 2.57 2.60 2.06 2.94 2.95 3.83 1.86 1.45 0.45 }
{ 3.62 3.28 1.82 0.43 3.48 3.05 0.16 1.67 3.74 0.27 0.66 0.85 0.47 2.36 0.47 2.47 2.32 2.76 2.93 3.66 2.68 1.19 3.24 1.51 3.99 3.09 0.82 0.68 1.55 0.72 2.32 0.78 }
{ 3.84 0.76 1.53 1.22 0.48 0.79 0.18 3.08 3.04 3.52 0.01 1.79 1.71 3.21 1.85 1.98 3.00 0.95 2.24 0.37 3.69 3.58 3.43 0.24 1.96 2.53 2.99 3.02 1.02 3.64 3.17 0.80 }
{ 2.21 0.20 2.09 3.16 2.82 0.10 2.93 2.21 3.66 3.50 3.19 1.22 1.58 1.93 0.21 2.61 0.00 3.07 3.02 0.88 0.62 0.23 2.68 0.13 1.38 2.79 3.19 1.75 3.93 2.54 1.18 1.91 }
{ 3.50 0.32 0.40 0.57 0.57 3.67 1.23 1.50 3.26 0.19 3.26 0.96 3.59 1.03 1.08 0.83 0.92 3.38 3.50 3.55 3.38 2.44 0.16 0.69 0.74 1.42 1.80 0.74 3.58 2.49 2.19 3.65 }
```

Matrice B :



```
Matrix B:
{ 2.09 2.92 4.00 3.88 3.47 3.36 0.59 1.50 1.14 0.68 3.90 2.97 2.87 0.12 2.58 3.25 2.94 3.27 0.50 2.88 1.50 2.60 3.66 3.56 0.13 3.09 1.80 0.03 0.91 1.67 0.63 3.68 }
{ 2.01 0.97 3.02 3.63 0.85 0.13 3.08 0.91 2.23 3.23 3.41 1.33 2.16 3.97 3.48 2.59 0.91 3.25 1.48 1.47 0.14 2.25 1.59 0.48 1.30 3.60 0.81 3.76 0.50 3.40 3.58 1.78 }
{ 0.89 2.54 2.24 2.50 1.69 0.07 3.46 0.56 1.07 1.17 2.73 2.25 3.84 2.43 3.15 1.69 3.27 2.96 0.54 2.07 0.08 0.03 2.99 1.34 3.61 1.85 3.07 0.20 2.52 0.10 2.73 3.88 }
{ 2.00 2.83 1.06 3.29 2.59 2.39 1.05 3.38 1.84 2.99 1.41 0.06 1.91 2.51 1.62 1.53 0.11 1.16 2.39 3.54 1.73 3.73 1.53 2.13 0.04 3.90 3.46 1.75 2.80 1.33 2.41 1.99 }
{ 2.24 0.16 1.51 0.93 3.75 3.44 3.86 2.44 2.34 0.31 1.63 1.22 1.99 0.32 3.50 1.22 3.13 3.73 2.35 0.86 1.12 2.24 3.50 1.00 1.21 2.91 3.47 0.92 0.76 1.79 3.56 2.30 }
{ 0.12 3.98 0.24 2.48 1.03 0.68 0.46 0.93 1.98 3.84 0.20 0.32 1.35 1.12 2.06 1.96 1.65 3.70 2.40 1.17 1.90 1.61 3.81 2.19 3.34 3.18 3.21 2.37 3.48 1.27 3.76 1.07 }
{ 3.77 2.88 0.51 3.10 1.89 0.70 0.11 0.17 1.82 3.45 0.54 0.06 0.85 3.27 3.19 3.12 2.63 1.19 3.13 3.69 3.37 0.97 2.00 0.09 0.30 0.12 1.42 2.86 1.64 0.72 1.19 0.73 }
{ 2.98 0.54 0.23 2.10 1.44 3.23 0.64 3.60 0.53 0.12 0.73 3.27 1.00 2.28 1.39 2.16 2.66 3.48 3.73 3.78 0.42 0.61 2.23 3.20 3.58 0.29 3.49 3.29 1.55 2.20 2.69 1.80 }
{ 1.13 3.80 3.93 2.91 2.38 1.09 0.53 0.79 2.40 1.95 1.42 0.92 1.91 2.77 3.79 2.68 3.93 0.95 2.05 0.94 3.59 1.94 1.31 2.15 2.28 1.35 1.39 0.68 1.16 0.88 3.69 2.96 }
{ 3.23 2.60 2.75 1.08 3.07 0.26 2.89 3.44 1.37 0.18 1.94 0.08 3.35 2.47 2.16 2.42 2.27 3.11 2.56 2.79 3.04 3.47 0.63 1.69 3.15 0.09 1.75 0.17 2.03 1.08 0.27 0.98 }
{ 3.93 2.03 2.46 0.28 1.47 3.82 3.44 1.35 0.97 1.16 0.20 3.97 0.69 0.28 0.88 1.20 3.15 2.60 0.33 1.05 0.55 0.87 2.25 0.71 2.53 0.43 1.81 3.04 1.34 0.23 1.13 2.78 }
{ 3.49 2.27 3.47 0.32 3.18 4.00 2.54 2.48 2.38 2.37 0.78 3.92 0.93 0.17 0.96 1.02 1.63 3.18 1.09 3.60 1.52 2.91 0.83 0.21 2.42 1.42 1.44 2.67 3.90 0.89 3.58 1.39 }
{ 3.81 2.66 0.81 1.13 1.42 2.13 1.16 0.88 0.40 0.43 2.84 3.38 2.69 3.28 0.02 1.97 2.42 3.74 1.16 2.42 2.99 3.50 3.65 3.32 2.42 1.22 3.14 0.60 2.17 2.76 1.58 3.97 }
{ 0.41 1.65 0.11 2.75 1.04 0.68 0.91 0.76 3.45 3.05 0.19 1.93 0.22 3.84 0.34 2.26 3.85 2.30 3.51 1.63 1.61 1.65 2.63 3.29 3.78 1.13 0.01 2.47 0.20 2.62 1.69 3.82 }
{ 2.42 0.80 1.66 1.36 2.49 3.23 0.02 0.42 1.58 1.11 1.27 3.30 3.41 1.25 0.79 2.26 1.88 3.98 1.73 1.47 1.68 1.54 2.32 3.51 1.35 1.57 3.80 2.04 1.67 2.43 3.11 0.33 }
{ 0.60 0.83 0.35 1.27 1.89 2.52 1.33 0.32 2.46 0.57 1.12 3.30 1.76 3.98 0.88 0.43 2.08 1.06 0.03 2.31 3.08 0.43 3.88 1.48 3.62 2.09 0.89 1.21 1.91 1.24 1.66 3.06 }
{ 3.37 1.56 1.71 2.90 2.94 0.37 0.33 3.01 3.39 2.55 1.44 1.90 0.38 1.66 0.75 1.37 3.65 0.92 1.24 0.62 1.37 0.73 1.09 0.82 0.69 1.34 0.26 2.01 3.85 3.24 1.79 3.65 }
{ 0.72 1.93 1.89 2.80 0.99 3.75 1.11 1.20 3.29 3.28 2.52 3.18 0.03 1.04 1.16 3.81 3.44 2.07 2.83 1.02 3.68 1.97 0.37 3.30 0.31 1.13 1.54 2.57 2.06 1.73 0.56 3.90 }
{ 0.30 0.02 3.66 1.65 3.02 0.75 0.41 1.69 2.71 2.11 3.30 3.35 1.09 2.00 3.48 0.60 3.88 2.15 0.88 0.27 0.82 0.79 0.77 0.80 3.83 2.37 2.96 1.18 2.72 3.14 3.69 3.71 }
{ 3.38 1.04 0.42 0.25 3.62 0.59 3.57 1.44 3.83 1.53 1.33 2.74 3.55 1.52 1.73 2.70 1.07 1.93 2.11 3.21 3.18 2.99 1.95 3.31 2.24 1.12 1.39 0.22 3.65 3.05 1.47 1.13 }
{ 0.84 1.30 0.20 3.53 1.00 1.62 0.58 3.50 2.43 3.13 0.79 1.81 3.05 1.73 2.55 3.22 0.70 1.29 3.85 2.84 0.79 3.45 0.19 2.97 0.14 0.52 0.33 2.21 2.28 2.13 0.88 0.31 }
{ 1.22 0.26 1.65 0.53 1.01 3.28 2.04 1.07 3.95 3.67 0.49 1.14 2.83 1.88 1.18 1.38 2.10 3.24 2.07 0.27 3.32 2.22 0.41 2.04 3.79 2.89 1.90 1.81 3.68 2.20 1.46 3.54 }
{ 2.64 0.49 2.03 0.45 1.12 2.19 3.17 1.48 3.43 2.98 1.86 1.44 1.77 2.34 1.52 2.53 2.23 1.93 3.35 0.74 0.04 0.44 0.16 0.89 1.02 1.64 2.83 1.77 0.53 2.18 3.43 3.23 }
{ 0.87 3.70 1.25 0.57 2.43 0.04 3.82 2.15 1.55 0.05 0.95 3.54 2.37 0.61 0.07 2.14 3.67 2.61 0.90 1.62 3.86 0.39 1.60 2.02 1.34 1.31 2.50 1.00 1.16 2.13 3.59 1.91 }
{ 1.31 2.70 1.84 2.33 1.65 3.69 0.48 2.66 0.59 0.99 0.45 0.41 2.76 3.31 1.96 1.81 2.61 1.67 2.56 0.40 3.32 1.47 3.51 2.08 3.85 1.58 2.90 2.52 1.50 3.25 3.41 0.26 }
{ 2.97 1.45 0.94 3.28 3.13 1.10 2.25 0.70 0.50 3.03 1.66 1.12 0.68 2.18 0.96 1.27 0.90 2.63 2.79 2.68 2.57 0.22 1.32 3.81 1.28 1.23 0.65 1.47 1.11 3.84 0.94 1.04 }
{ 2.50 3.02 2.33 1.08 2.85 1.35 2.86 0.09 2.80 2.52 0.51 2.88 1.52 2.10 3.46 2.37 1.98 0.12 2.69 2.21 1.40 2.29 3.06 0.09 0.00 2.19 2.99 1.13 3.18 3.34 2.90 1.19 }
{ 2.15 1.62 0.41 1.56 0.39 2.67 0.65 2.76 2.68 2.95 3.55 3.03 2.28 1.06 0.45 1.07 2.93 3.91 1.95 0.86 3.20 2.45 1.39 1.93 3.99 1.49 3.79 0.88 2.52 2.30 0.58 3.72 }
{ 1.83 2.57 0.19 2.62 2.97 0.40 2.60 0.80 0.94 2.93 3.98 3.02 3.57 2.19 3.57 3.48 2.46 2.88 2.92 0.61 3.02 0.61 1.62 2.46 2.87 3.32 3.54 0.25 1.58 2.41 1.95 1.81 }
{ 3.05 2.32 1.34 2.33 1.44 3.00 0.33 0.38 3.99 0.18 1.59 0.43 1.66 1.74 1.08 1.81 0.90 1.05 2.57 3.57 3.52 1.80 1.65 0.93 3.86 2.96 1.37 2.89 2.41 2.49 3.58 2.73 }
{ 0.94 0.44 2.13 2.86 2.27 0.06 2.04 1.83 3.88 3.56 2.81 0.98 0.41 1.07 1.87 2.43 0.20 1.54 0.47 3.99 2.96 0.16 0.02 2.72 3.83 2.83 3.00 0.56 2.71 2.45 2.65 3.90 }
{ 0.28 3.63 3.45 2.51 3.56 2.30 0.61 3.59 2.30 1.05 0.70 2.50 3.34 2.68 3.43 1.79 3.93 2.64 2.84 0.23 3.26 2.99 3.90 3.34 1.26 0.47 0.82 1.17 3.79 3.37 1.00 0.86 }
```

Matrice C :



```
Matrix C:
{ 115.31 112.85 89.07 117.14 114.76 107.13 85.01 103.48 118.60 113.65 86.02 106.86 110.85 114.08 111.23 117.48 121.38 137.25 133.00 116.29 117.33 119.75 100.92 114.01 117.46 94.55 117.04 96.14 121.42 114.85 121.94 115.02 }
{ 145.91 125.33 123.53 123.67 153.73 123.83 122.57 116.21 157.91 145.27 99.26 136.91 128.82 120.27 134.54 138.91 152.49 156.30 134.98 130.46 136.59 128.05 116.70 123.72 136.33 121.41 143.00 109.04 155.78 138.50 157.24 150.29 }
{ 137.50 124.73 101.68 139.33 137.33 118.06 103.31 107.35 143.07 145.36 105.93 141.69 123.67 137.75 128.79 135.89 162.62 168.40 148.39 118.06 138.02 115.35 131.66 133.07 148.53 111.74 139.36 121.85 138.59 151.87 147.46 143.69 }
{ 117.70 135.91 93.46 144.23 126.97 118.97 90.21 110.35 138.83 125.60 105.08 133.36 125.50 136.56 112.10 134.66 156.71 152.45 133.65 126.10 149.87 108.18 136.62 146.27 159.53 116.07 138.83 104.53 141.93 137.78 141.21 162.83 }
{ 124.71 127.36 116.30 125.51 132.06 113.46 102.95 102.64 146.86 121.40 105.57 126.02 132.23 125.80 122.11 127.20 158.39 155.82 125.88 118.01 146.89 110.66 119.78 121.26 163.84 112.01 143.75 101.46 139.16 135.20 157.31 150.81 }
{ 118.91 110.49 97.77 117.59 137.56 102.87 96.66 96.45 114.12 109.36 93.08 114.28 108.44 114.03 111.85 110.64 136.19 130.36 109.50 111.78 122.92 98.25 114.76 108.32 114.95 93.45 108.71 85.62 126.15 114.55 112.97 122.24 }
{ 125.17 116.00 107.08 125.76 134.57 114.03 117.36 92.14 139.17 130.82 117.34 145.06 121.34 119.49 111.22 134.09 154.40 158.62 111.29 116.33 148.12 93.73 115.26 129.92 134.35 116.21 131.48 96.42 129.19 129.93 132.01 162.99 }
{ 139.36 135.76 121.47 128.48 150.65 124.34 128.31 106.99 151.74 129.83 109.31 148.63 132.68 135.28 132.00 146.04 164.68 167.47 139.92 129.94 149.82 118.47 135.30 136.03 154.19 118.00 144.40 115.04 140.66 146.04 162.67 152.16 }
{ 127.27 136.15 109.33 125.07 143.82 113.29 109.35 103.94 132.05 111.94 101.98 116.12 119.25 118.35 118.43 127.01 152.99 139.98 120.58 119.33 142.56 107.70 126.45 113.02 118.87 112.52 129.78 90.66 127.42 117.40 137.23 143.57 }
{ 137.20 131.54 102.76 134.00 140.31 136.68 119.45 113.76 149.81 124.63 104.62 142.12 126.18 135.13 118.36 134.46 161.32 160.91 136.55 136.27 132.92 111.43 150.36 131.94 154.39 130.19 151.38 115.28 138.51 137.53 160.72 170.42 }
{ 128.43 126.64 118.55 132.23 129.24 119.91 104.44 106.77 158.03 132.62 107.33 134.40 117.87 123.04 106.58 134.33 159.06 158.94 125.99 122.37 139.45 105.99 115.59 135.25 151.73 115.87 131.19 109.83 139.15 133.75 146.56 173.39 }
{ 115.78 95.16 98.73 116.85 132.23 104.03 102.51 86.55 138.20 122.25 100.03 124.97 113.30 114.45 106.63 119.29 135.06 152.53 112.44 105.95 129.11 100.41 112.06 126.65 132.61 114.42 118.13 91.29 119.66 135.29 125.85 141.63 }
{ 105.01 119.91 84.80 120.02 109.41 102.45 82.32 88.82 108.67 116.01 79.72 108.74 100.78 112.77 106.51 123.60 143.58 133.41 124.29 102.37 121.27 89.33 107.06 110.97 125.97 85.47 112.18 98.55 111.75 99.98 119.38 120.75 }
{ 126.07 126.63 102.76 134.81 143.06 108.45 113.17 105.82 143.73 127.10 95.33 127.34 129.99 139.44 128.54 134.59 162.04 162.70 143.38 119.40 142.01 116.38 139.23 138.08 144.25 118.75 134.13 112.12 134.05 152.08 156.22 143.54 }
{ 135.73 128.10 93.73 141.27 129.54 122.70 105.87 112.69 142.78 129.07 106.39 129.41 126.93 145.22 122.40 136.55 157.18 159.79 146.78 134.21 145.02 117.77 136.32 131.27 158.38 107.08 132.52 116.15 136.32 139.31 145.38 150.03 }
{ 132.54 114.52 107.22 108.49 138.66 117.31 127.79 104.42 141.61 108.38 101.70 136.71 120.46 115.56 117.58 116.76 143.59 155.63 120.74 130.66 131.10 109.98 128.17 115.99 155.76 105.88 132.94 94.77 139.57 128.37 135.21 145.20 }
{ 127.40 142.08 115.55 142.75 135.85 110.81 91.86 107.76 130.05 122.76 102.00 118.30 127.00 121.88 128.14 132.18 148.32 143.21 128.04 123.46 131.35 114.66 128.27 124.25 128.90 107.94 127.06 98.63 136.87 126.59 134.72 137.24 }
{ 144.17 134.65 118.85 145.74 153.62 130.84 122.08 123.39 165.20 150.52 124.31 147.12 133.55 129.32 134.08 145.93 164.51 169.61 145.24 141.69 144.72 114.51 127.72 141.78 166.10 135.90 162.37 114.53 161.98 145.97 166.77 176.58 }
{ 112.68 116.80 85.11 98.47 115.06 105.81 99.08 98.05 104.26 92.05 85.79 116.21 115.98 103.21 100.24 107.59 125.12 134.67 111.79 105.89 118.74 102.20 114.65 108.35 125.70 96.60 132.72 84.37 121.89 112.22 122.48 116.23 }
{ 127.18 127.20 104.51 128.11 131.95 123.91 108.96 108.42 143.11 128.00 105.19 126.24 127.48 133.17 128.26 124.91 143.43 151.65 138.22 125.04 137.41 114.68 124.34 107.20 137.71 126.96 146.16 114.69 137.89 135.96 159.67 137.95 }
{ 123.85 119.20 110.06 123.90 129.25 121.29 98.30 98.23 136.19 129.67 92.60 137.94 107.31 116.32 100.08 121.31 145.96 143.82 116.07 117.35 123.90 104.03 117.95 119.42 124.69 104.67 115.71 109.21 131.73 131.67 130.85 143.85 }
{ 147.28 148.76 136.43 152.02 160.34 142.36 121.92 117.32 173.51 143.81 117.27 157.23 135.16 142.14 138.40 153.92 187.51 173.72 141.08 149.99 163.27 123.79 148.53 136.77 169.46 128.95 152.90 127.81 158.00 146.88 172.34 180.29 }
{ 110.03 108.20 91.62 121.83 117.11 87.13 89.32 78.00 131.55 125.33 88.40 94.78 101.56 121.26 103.93 123.40 122.88 126.30 119.18 115.16 123.21 93.01 100.49 116.85 106.80 97.18 109.50 92.45 110.41 113.53 121.93 126.82 }
{ 145.58 137.25 117.24 151.22 164.42 140.16 121.38 125.44 172.00 149.90 123.31 152.52 151.94 145.41 144.73 159.75 166.00 177.93 165.71 152.47 177.86 132.26 135.75 159.55 167.38 137.12 168.01 119.86 160.32 171.58 170.72 159.47 }
{ 101.72 110.99 80.86 103.40 111.83 101.80 87.45 94.69 118.57 91.89 83.07 110.70 114.84 107.51 96.20 116.25 124.91 121.04 122.91 102.87 116.34 104.52 106.53 119.35 109.05 98.76 111.92 79.09 105.51 114.14 121.19 119.42 }
{ 138.23 129.19 117.15 143.39 142.31 137.94 114.57 128.46 167.44 145.45 117.07 147.79 136.77 153.31 129.19 144.37 172.88 167.29 153.16 128.30 153.29 134.11 134.01 148.04 157.91 124.05 150.35 120.48 158.37 160.84 150.07 180.24 }
{ 146.08 132.38 125.47 144.20 148.91 129.28 120.13 117.49 162.78 154.76 114.92 133.85 137.58 144.05 141.60 145.24 163.53 169.50 136.58 138.31 146.26 126.67 135.47 124.99 157.65 131.61 149.41 116.34 154.65 138.23 158.62 168.37 }
{ 122.87 118.15 91.43 120.75 122.71 123.87 98.26 94.05 129.51 127.72 99.24 128.66 124.72 119.73 111.11 131.70 141.20 144.89 134.44 113.60 134.65 103.22 115.04 126.60 128.71 111.46 143.84 99.01 118.25 129.60 134.15 137.53 }
{ 108.37 111.48 111.42 137.49 136.18 110.37 107.36 100.80 143.45 129.83 109.28 123.12 119.39 125.37 130.01 133.69 150.66 146.33 128.26 108.22 127.87 100.24 123.47 133.85 135.72 120.62 125.11 99.06 120.02 142.34 150.84 149.32 }
{ 128.66 115.91 115.38 138.74 133.69 123.72 92.72 112.07 155.05 131.99 111.71 126.48 125.62 132.33 117.57 134.86 149.28 149.17 140.22 131.33 137.12 115.99 111.58 136.72 152.31 116.51 133.04 101.31 137.74 145.93 140.65 163.61 }
{ 130.82 126.52 112.57 128.29 141.79 120.70 107.73 97.36 130.01 118.06 110.84 127.10 117.07 129.87 133.42 131.36 155.77 139.62 134.76 122.15 137.53 106.49 121.91 119.60 134.19 108.67 138.29 92.90 120.56 123.40 132.56 150.05 }
{ 111.85 125.86 105.50 127.08 131.94 114.84 86.41 92.82 135.16 123.65 103.12 136.64 120.40 109.80 122.60 133.95 145.98 143.53 123.27 109.96 141.23 113.44 120.99 138.88 130.90 107.40 124.30 91.19 139.37 131.61 124.52 142.84 }
```

Rangée :

```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_ROW);
```

Matrice A :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }
{ 3.64 1.82 3.69 2.66 0.60 2.55 2.28 1.69 3.77 2.16 2.31 2.15 1.02 1.59 1.40 0.15 3.18 0.79 0.28 1.56 2.36 0.28 0.79 0.62 2.58 1.82 2.42 2.79 1.77 2.74 1.59 3.34 }
{ 2.37 0.80 3.80 3.50 1.57 3.95 0.74 3.58 2.30 1.77 2.51 2.83 0.20 1.14 1.04 1.63 3.58 2.84 2.91 3.58 1.57 1.59 3.62 1.23 1.55 2.28 1.41 2.93 2.98 2.95 2.96 0.56 }
{ 0.80 1.09 2.72 3.64 1.47 2.07 0.44 3.63 0.81 2.07 2.62 1.75 2.75 0.36 0.32 0.30 0.11 1.42 1.23 2.79 0.57 1.58 0.27 2.70 2.90 0.79 2.78 2.46 2.62 2.18 0.39 2.18 }
{ 0.20 3.90 1.86 3.88 2.91 2.12 2.72 2.77 1.49 1.55 0.20 3.24 0.68 0.24 1.24 3.14 0.85 1.29 1.94 0.04 1.76 2.03 2.42 3.03 1.60 1.40 3.37 3.77 2.49 3.99 0.21 2.25 }
{ 3.45 1.55 1.23 1.14 0.11 2.45 1.56 0.07 0.12 1.82 3.43 3.75 1.61 2.91 2.45 3.23 3.50 2.33 2.38 0.32 3.17 1.85 2.36 2.36 2.28 3.26 2.69 1.37 0.04 1.07 1.37 0.93 }
{ 3.08 1.49 3.48 0.47 2.90 2.52 3.58 0.73 2.14 3.61 3.12 3.86 0.79 3.61 3.19 3.81 3.40 3.56 2.51 0.74 2.47 1.06 1.00 2.93 3.15 0.68 2.86 1.92 0.33 3.23 3.42 0.50 }
{ 1.23 2.29 1.89 3.05 0.77 2.91 3.98 0.06 2.59 2.77 0.70 0.29 1.10 2.71 3.36 2.23 1.48 3.12 0.93 2.55 0.93 0.82 3.88 1.13 0.09 3.08 0.61 0.19 0.04 1.39 2.58 1.37 }
{ 1.66 1.67 0.36 2.18 2.80 1.35 2.63 3.50 2.14 2.28 0.22 1.86 1.04 1.71 3.45 1.95 0.35 2.26 2.84 3.11 3.46 3.40 1.17 3.46 2.58 2.99 3.31 3.06 3.49 3.63 3.65 2.09 }
{ 1.76 0.43 1.42 3.29 1.71 0.68 0.00 2.63 3.09 0.25 0.36 1.54 2.74 3.23 0.62 0.57 0.19 1.38 0.07 3.91 2.59 0.21 3.66 2.55 1.73 0.08 0.97 1.57 2.62 3.13 0.09 3.30 }
{ 0.55 2.54 2.20 3.93 1.85 0.86 1.20 3.13 1.37 1.92 2.43 0.41 3.94 3.02 0.60 1.83 2.85 3.99 3.10 2.57 2.48 3.78 2.75 1.16 2.27 1.16 2.19 2.85 0.99 2.19 1.97 3.81 }
{ 2.68 3.06 2.49 1.78 3.20 2.98 3.96 0.82 2.52 3.64 2.37 3.67 2.02 1.96 1.80 3.56 3.76 0.73 1.01 0.81 2.92 3.76 2.50 0.17 2.57 0.14 2.23 2.18 1.79 0.25 3.98 0.51 }
{ 2.40 0.20 0.69 2.64 1.33 1.26 2.61 1.64 1.71 0.89 2.71 0.67 1.99 2.03 3.21 1.74 0.02 1.36 1.50 0.51 3.31 2.93 3.56 2.57 2.60 2.06 2.94 2.95 3.83 1.86 1.45 0.45 }
{ 3.62 3.28 1.82 0.43 3.48 3.05 0.16 1.67 3.74 0.27 0.66 0.85 0.47 2.36 0.47 2.47 2.32 2.76 2.93 3.66 2.68 1.19 3.24 1.51 3.99 3.09 0.82 0.68 1.55 0.72 2.32 0.78 }
{ 3.84 0.76 1.53 1.22 0.48 0.79 0.18 3.08 3.04 3.52 0.01 1.79 1.71 3.21 1.85 1.98 3.00 0.95 2.24 0.37 3.69 3.58 3.43 0.24 1.96 2.53 2.99 3.02 1.02 3.64 3.17 0.80 }
{ 2.21 0.20 2.09 3.16 2.82 0.10 2.93 2.21 3.66 3.50 3.19 1.22 1.58 1.93 0.21 2.61 0.00 3.07 3.02 0.88 0.62 0.23 2.68 0.13 1.38 2.79 3.19 1.75 3.93 2.54 1.18 1.91 }
{ 3.50 0.32 0.40 0.57 0.57 3.67 1.23 1.50 3.26 0.19 3.26 0.96 3.59 1.03 1.08 0.83 0.92 3.38 3.50 3.55 3.38 2.44 0.16 0.69 0.74 1.42 1.80 0.74 3.58 2.49 2.19 3.65 }
```

Matrice B :

```
Matrix B:
{ 2.09 2.92 4.00 3.88 3.47 3.36 0.59 1.50 1.14 0.68 3.90 2.97 2.87 0.12 2.58 3.25 2.94 3.27 0.50 2.88 1.50 2.60 3.66 3.56 0.13 3.09 1.80 0.03 0.91 1.67 0.63 3.68 }
{ 2.01 0.97 3.02 3.63 0.85 0.13 3.08 0.91 2.23 3.23 3.41 1.33 2.16 3.97 3.48 2.59 0.91 3.25 1.48 1.47 0.14 2.25 1.59 0.48 1.30 3.60 0.81 3.76 0.50 3.40 3.58 1.78 }
{ 0.89 2.54 2.24 2.50 1.69 0.07 3.46 0.56 1.07 1.17 2.73 2.25 3.84 2.43 3.15 1.69 3.27 2.96 0.54 2.07 0.08 0.03 2.99 1.34 3.61 1.85 3.07 0.20 2.52 0.10 2.73 3.88 }
{ 2.00 2.83 1.06 3.29 2.59 2.39 1.05 3.38 1.84 2.99 1.41 0.06 1.91 2.51 1.62 1.53 0.11 1.16 2.39 3.54 1.73 3.73 1.53 2.13 0.04 3.90 3.46 1.75 2.80 1.33 2.41 1.99 }
{ 2.24 0.16 1.51 0.93 3.75 3.44 3.86 2.44 2.34 0.31 1.63 1.22 1.99 0.32 3.50 1.22 3.13 3.73 2.35 0.86 1.12 2.24 3.50 1.00 1.21 2.91 3.47 0.92 0.76 1.79 3.56 2.30 }
{ 0.12 3.98 0.24 2.48 1.03 0.68 0.46 0.93 1.98 3.84 0.20 0.32 1.35 1.12 2.06 1.96 1.65 3.70 2.40 1.17 1.90 1.61 3.81 2.19 3.34 3.18 3.21 2.37 3.48 1.27 3.76 1.07 }
{ 3.77 2.88 0.51 3.10 1.89 0.70 0.11 0.17 1.82 3.45 0.54 0.06 0.85 3.27 3.19 3.12 2.63 1.19 3.13 3.69 3.37 0.97 2.00 0.09 0.30 0.12 1.42 2.86 1.64 0.72 1.19 0.73 }
{ 2.98 0.54 0.23 2.10 1.44 3.23 0.64 3.60 0.53 0.12 0.73 3.27 1.00 2.28 1.39 2.16 2.66 3.48 3.73 3.78 0.42 0.61 2.23 3.20 3.58 0.29 3.49 3.29 1.55 2.20 2.69 1.80 }
{ 1.13 3.80 3.93 2.91 2.38 1.09 0.53 0.79 2.40 1.95 1.42 0.92 1.91 2.77 3.79 2.68 3.93 0.95 2.05 0.94 3.59 1.94 1.31 2.15 2.28 1.35 1.39 0.68 1.16 0.88 3.69 2.96 }
{ 3.23 2.60 2.75 1.08 3.07 0.26 2.89 3.44 1.37 0.18 1.94 0.08 3.35 2.47 2.16 2.42 2.27 3.11 2.56 2.79 3.04 3.47 0.63 1.69 3.15 0.09 1.75 0.17 2.03 1.08 0.27 0.98 }
{ 3.93 2.03 2.46 0.28 1.47 3.82 3.44 1.35 0.97 1.16 0.20 3.97 0.69 0.28 0.88 1.20 3.15 2.60 0.33 1.05 0.55 0.87 2.25 0.71 2.53 0.43 1.81 3.04 1.34 0.23 1.13 2.78 }
{ 3.49 2.27 3.47 0.32 3.18 4.00 2.54 2.48 2.38 2.37 0.78 3.92 0.93 0.17 0.96 1.02 1.63 3.18 1.09 3.60 1.52 2.91 0.83 0.21 2.42 1.42 1.44 2.67 3.90 0.89 3.58 1.39 }
{ 3.81 2.66 0.81 1.13 1.42 2.13 1.16 0.88 0.40 0.43 2.84 3.38 2.69 3.28 0.02 1.97 2.42 3.74 1.16 2.42 2.99 3.50 3.65 3.32 2.42 1.22 3.14 0.60 2.17 2.76 1.58 3.97 }
{ 0.41 1.65 0.11 2.75 1.04 0.68 0.91 0.76 3.45 3.05 0.19 1.93 0.22 3.84 0.34 2.26 3.85 2.30 3.51 1.63 1.61 1.65 2.63 3.29 3.78 1.13 0.01 2.47 0.20 2.62 1.69 3.82 }
{ 2.42 0.80 1.66 1.36 2.49 3.23 0.02 0.42 1.58 1.11 1.27 3.30 3.41 1.25 0.79 2.26 1.88 3.98 1.73 1.47 1.68 1.54 2.32 3.51 1.35 1.57 3.80 2.04 1.67 2.43 3.11 0.33 }
{ 0.60 0.83 0.35 1.27 1.89 2.52 1.33 0.32 2.46 0.57 1.12 3.30 1.76 3.98 0.88 0.43 2.08 1.06 0.03 2.31 3.08 0.43 3.88 1.48 3.62 2.09 0.89 1.21 1.91 1.24 1.66 3.06 }
{ 3.37 1.56 1.71 2.90 2.94 0.37 0.33 3.01 3.39 2.55 1.44 1.90 0.38 1.66 0.75 1.37 3.65 0.92 1.24 0.62 1.37 0.73 1.09 0.82 0.69 1.34 0.26 2.01 3.85 3.24 1.79 3.65 }
{ 0.72 1.93 1.89 2.80 0.99 3.75 1.11 1.20 3.29 3.28 2.52 3.18 0.03 1.04 1.16 3.81 3.44 2.07 2.83 1.02 3.68 1.97 0.37 3.30 0.31 1.13 1.54 2.57 2.06 1.73 0.56 3.90 }
{ 0.30 0.02 3.66 1.65 3.02 0.75 0.41 1.69 2.71 2.11 3.30 3.35 1.09 2.00 3.48 0.60 3.88 2.15 0.88 0.27 0.82 0.79 0.77 0.80 3.83 2.37 2.96 1.18 2.72 3.14 3.69 3.71 }
{ 3.38 1.04 0.42 0.25 3.62 0.59 3.57 1.44 3.83 1.53 1.33 2.74 3.55 1.52 1.73 2.70 1.07 1.93 2.11 3.21 3.18 2.99 1.95 3.31 2.24 1.12 1.39 0.22 3.65 3.05 1.47 1.13 }
{ 0.84 1.30 0.20 3.53 1.00 1.62 0.58 3.50 2.43 3.13 0.79 1.81 3.05 1.73 2.55 3.22 0.70 1.29 3.85 2.84 0.79 3.45 0.19 2.97 0.14 0.52 0.33 2.21 2.28 2.13 0.88 0.31 }
{ 1.22 0.26 1.65 0.53 1.01 3.28 2.04 1.07 3.95 3.67 0.49 1.14 2.83 1.88 1.18 1.38 2.10 3.24 2.07 0.27 3.32 2.22 0.41 2.04 3.79 2.89 1.90 1.81 3.68 2.20 1.46 3.54 }
{ 2.64 0.49 2.03 0.45 1.12 2.19 3.17 1.48 3.43 2.98 1.86 1.44 1.77 2.34 1.52 2.53 2.23 1.93 3.35 0.74 0.04 0.44 0.16 0.89 1.02 1.64 2.83 1.77 0.53 2.18 3.43 3.23 }
{ 0.87 3.70 1.25 0.57 2.43 0.04 3.82 2.15 1.55 0.05 0.95 3.54 2.37 0.61 0.07 2.14 3.67 2.61 0.90 1.62 3.86 0.39 1.60 2.02 1.34 1.31 2.50 1.00 1.16 2.13 3.59 1.91 }
{ 1.31 2.70 1.84 2.33 1.65 3.69 0.48 2.66 0.59 0.99 0.45 0.41 2.76 3.31 1.96 1.81 2.61 1.67 2.56 0.40 3.32 1.47 3.51 2.08 3.85 1.58 2.90 2.52 1.50 3.25 3.41 0.26 }
{ 2.97 1.45 0.94 3.28 3.13 1.10 2.25 0.70 0.50 3.03 1.66 1.12 0.68 2.18 0.96 1.27 0.90 2.63 2.79 2.68 2.57 0.22 1.32 3.81 1.28 1.23 0.65 1.47 1.11 3.84 0.94 1.04 }
{ 2.50 3.02 2.33 1.08 2.85 1.35 2.86 0.09 2.80 2.52 0.51 2.88 1.52 2.10 3.46 2.37 1.98 0.12 2.69 2.21 1.40 2.29 3.06 0.09 0.00 2.19 2.99 1.13 3.18 3.34 2.90 1.19 }
{ 2.15 1.62 0.41 1.56 0.39 2.67 0.65 2.76 2.68 2.95 3.55 3.03 2.28 1.06 0.45 1.07 2.93 3.91 1.95 0.86 3.20 2.45 1.39 1.93 3.99 1.49 3.79 0.88 2.52 2.30 0.58 3.72 }
{ 1.83 2.57 0.19 2.62 2.97 0.40 2.60 0.80 0.94 2.93 3.98 3.02 3.57 2.19 3.57 3.48 2.46 2.88 2.92 0.61 3.02 0.61 1.62 2.46 2.87 3.32 3.54 0.25 1.58 2.41 1.95 1.81 }
{ 3.05 2.32 1.34 2.33 1.44 3.00 0.33 0.38 3.99 0.18 1.59 0.43 1.66 1.74 1.08 1.81 0.90 1.05 2.57 3.57 3.52 1.80 1.65 0.93 3.86 2.96 1.37 2.89 2.41 2.49 3.58 2.73 }
{ 0.94 0.44 2.13 2.86 2.27 0.06 2.04 1.83 3.88 3.56 2.81 0.98 0.41 1.07 1.87 2.43 0.20 1.54 0.47 3.99 2.96 0.16 0.02 2.72 3.83 2.83 3.00 0.56 2.71 2.45 2.65 3.90 }
{ 0.28 3.63 3.45 2.51 3.56 2.30 0.61 3.59 2.30 1.05 0.70 2.50 3.34 2.68 3.43 1.79 3.93 2.64 2.84 0.23 3.26 2.99 3.90 3.34 1.26 0.47 0.82 1.17 3.79 3.37 1.00 0.86 }
```

Matrice C :

```
Matrix C:
{ 115.31 112.85 89.07 117.14 114.76 107.13 85.01 103.48 118.60 113.65 86.02 106.86 110.85 114.08 111.23 117.48 121.38 137.25 133.00 116.29 117.33 119.75 100.92 114.01 117.46 94.55 117.04 96.14 121.42 114.85 121.94 115.02 }
{ 145.91 125.33 123.53 123.67 153.73 123.83 122.57 116.21 157.91 145.27 99.26 136.91 128.82 120.27 134.54 138.91 152.49 156.30 134.98 130.46 136.59 128.05 116.70 123.72 136.33 121.41 143.00 109.04 155.78 138.50 157.24 150.29 }
{ 137.58 124.73 108.88 139.31 137.33 118.06 103.31 107.35 143.97 145.26 105.93 147.03 123.67 127.75 128.79 135.89 162.62 168.40 148.39 118.86 138.02 115.35 131.68 133.07 148.53 111.74 139.36 121.85 138.59 151.87 147.46 143.69 }
{ 117.70 135.91 93.46 144.23 126.97 118.97 90.21 110.35 138.83 125.60 105.08 133.36 125.50 136.56 112.10 134.66 156.71 152.45 133.65 126.10 149.87 108.18 136.62 146.27 159.53 116.07 138.83 104.53 141.93 137.78 141.21 162.83 }
{ 124.71 127.36 116.30 125.51 132.06 113.46 102.95 102.64 146.86 121.40 105.57 126.02 132.23 125.80 122.11 127.20 158.39 155.82 125.88 118.01 146.89 110.66 119.78 121.26 163.04 112.01 143.75 103.46 139.16 135.20 157.31 150.81 }
{ 118.91 110.49 97.77 117.59 137.56 102.87 96.66 96.45 114.12 109.36 93.08 114.28 108.44 114.03 111.85 110.64 136.19 130.36 109.50 111.78 122.92 98.25 114.76 108.32 114.95 93.45 108.71 85.62 126.15 114.55 112.97 122.24 }
{ 125.17 116.00 107.08 125.76 134.57 114.03 117.36 92.14 139.17 130.82 117.34 145.06 121.34 119.49 111.22 134.09 154.40 158.62 111.29 116.33 148.12 93.73 115.26 129.92 134.35 116.21 131.48 96.42 129.19 129.93 132.01 162.99 }
{ 139.36 135.70 121.47 128.48 150.65 124.34 128.31 106.99 151.74 129.83 109.31 148.63 132.68 135.28 132.00 140.04 164.68 167.47 139.92 129.94 149.82 118.47 135.30 136.03 154.19 118.00 144.40 115.04 140.66 146.04 162.67 152.16 }
{ 127.27 136.15 109.33 125.07 143.82 113.29 109.35 103.94 132.05 111.94 101.98 116.12 119.25 118.35 118.43 127.01 152.99 139.98 120.58 119.33 142.56 107.70 126.45 113.02 118.87 112.52 129.78 90.66 127.42 117.40 137.23 143.57 }
{ 137.20 131.54 102.76 134.00 140.31 136.68 119.45 113.76 149.81 124.63 104.62 142.12 126.18 135.13 118.36 134.46 161.32 160.91 136.55 136.27 132.92 111.43 150.36 131.94 154.39 130.19 151.38 115.28 138.51 137.53 160.72 170.42 }
{ 128.43 126.64 118.55 132.23 129.24 119.91 104.44 106.77 158.03 132.62 107.33 134.40 117.87 123.04 106.58 134.33 159.06 158.94 125.99 122.37 139.45 105.99 115.59 135.25 151.73 115.87 131.19 109.83 139.15 133.75 146.56 173.39 }
{ 115.78 95.16 98.73 116.85 132.23 104.03 102.51 86.55 138.20 122.25 109.03 124.97 113.30 114.45 106.63 119.29 135.06 152.53 112.44 105.95 129.11 100.41 112.06 126.65 132.61 114.42 118.13 91.29 119.66 135.29 125.85 141.63 }
{ 105.01 119.91 84.80 120.02 109.41 102.45 82.32 88.82 108.67 116.01 79.72 108.74 100.78 112.77 106.51 123.60 143.58 133.41 124.29 102.37 121.27 89.33 107.06 110.97 125.97 85.47 112.18 98.55 111.75 99.98 119.38 120.75 }
{ 126.07 126.63 102.76 134.81 143.06 108.45 113.17 105.82 143.73 127.10 95.33 127.34 129.99 139.44 128.54 134.59 162.04 162.70 143.38 119.40 142.01 116.38 139.23 138.08 144.25 118.75 134.13 112.12 134.05 152.08 156.22 143.54 }
{ 135.73 128.13 93.73 141.27 129.54 122.70 105.87 112.69 142.78 129.07 106.39 129.41 126.93 145.22 122.40 136.55 157.18 159.79 146.78 134.21 145.02 117.77 136.32 131.27 158.38 107.08 132.52 116.15 136.42 139.31 145.38 150.03 }
{ 132.54 114.52 107.22 108.49 138.66 117.31 127.79 104.42 141.61 108.38 101.70 136.71 120.46 115.56 117.58 116.76 143.59 155.63 120.74 130.66 131.10 109.98 128.17 115.99 155.76 105.88 132.94 94.77 139.57 128.37 135.21 145.20 }
{ 127.40 142.08 115.55 142.75 135.85 110.81 91.86 107.76 130.05 122.76 102.00 118.30 127.00 121.88 128.14 132.18 148.32 143.21 128.04 123.46 131.35 114.66 128.27 124.25 128.90 107.94 127.06 98.63 136.87 126.59 134.72 137.24 }
{ 144.17 134.65 118.85 145.74 153.62 130.84 122.08 123.39 165.20 150.52 124.31 147.12 133.55 129.32 134.88 145.93 164.51 169.61 145.24 141.69 144.72 114.51 127.72 141.78 166.10 135.90 162.37 114.53 161.98 145.97 166.77 176.58 }
{ 112.68 116.80 85.11 98.47 115.06 105.81 99.08 98.05 104.26 92.05 85.79 116.21 115.98 103.21 100.24 107.59 125.12 134.67 111.79 105.89 118.74 102.20 114.65 108.35 125.70 96.60 132.72 84.37 121.89 112.22 122.48 116.23 }
{ 127.18 127.20 104.51 128.11 131.95 123.91 108.96 108.42 143.11 128.00 105.19 126.24 127.48 133.17 128.26 124.91 143.43 151.65 138.22 125.04 137.41 114.68 124.34 107.20 137.71 126.96 146.16 114.69 137.89 135.96 159.67 137.95 }
{ 123.85 119.20 110.06 123.90 129.25 121.29 98.30 98.23 136.19 129.67 92.60 137.94 107.31 116.32 100.08 121.31 145.96 143.82 116.07 117.35 123.90 104.03 117.95 119.42 124.69 104.67 115.71 109.21 131.73 131.67 130.85 143.85 }
{ 147.28 148.76 136.43 152.02 160.34 142.36 121.92 117.32 173.51 143.81 117.27 157.23 135.16 142.14 138.40 153.92 187.51 173.72 141.08 149.99 163.27 123.79 148.53 136.77 169.46 128.95 152.90 127.81 158.00 146.88 172.34 180.29 }
{ 110.03 108.20 91.62 121.83 117.11 87.13 89.32 78.00 131.55 125.33 88.40 94.78 101.56 121.26 103.93 123.40 122.88 126.30 119.18 115.16 123.21 93.01 100.49 116.85 106.80 97.18 109.50 92.45 110.41 113.53 121.93 126.82 }
{ 145.58 137.25 117.24 151.22 164.42 140.16 121.38 125.44 172.00 149.90 123.31 152.52 151.94 145.44 144.73 159.75 166.00 177.93 165.71 152.47 177.86 132.26 135.75 159.55 167.38 138.12 168.01 119.86 160.32 171.58 170.72 159.47 }
{ 101.72 110.99 80.06 103.40 111.83 101.80 87.45 94.69 118.57 91.89 83.07 110.70 114.84 107.51 96.20 116.25 124.91 121.04 122.91 102.87 116.34 104.52 106.53 119.35 109.05 90.76 111.92 79.09 105.51 114.14 121.19 119.42 }
{ 138.23 129.19 117.15 143.39 142.31 137.94 114.57 128.46 167.44 145.45 117.07 147.79 136.77 153.31 129.19 144.37 172.88 167.29 153.16 128.30 153.29 134.11 134.01 148.04 157.91 124.05 150.35 120.48 158.37 160.84 150.07 180.24 }
{ 146.08 132.38 125.47 144.20 148.91 129.28 120.13 117.49 162.78 154.76 114.92 133.85 137.58 144.05 141.60 145.24 163.53 169.50 136.58 138.31 146.26 126.07 135.47 124.99 157.65 131.61 149.41 116.34 154.65 138.23 158.62 168.37 }
{ 122.87 118.15 91.43 120.75 122.71 123.87 98.20 94.05 129.51 127.72 99.24 128.66 124.72 119.73 111.11 131.70 141.20 144.89 134.44 113.60 134.65 103.22 115.04 126.60 128.71 111.46 143.84 99.01 118.25 129.60 134.15 137.53 }
{ 108.37 111.48 111.42 137.49 136.18 110.37 107.36 100.80 143.45 129.83 109.28 123.12 119.39 125.37 130.01 133.69 150.66 146.33 128.26 108.22 127.87 100.24 123.47 133.85 135.72 120.62 125.11 99.06 120.02 142.34 150.84 149.32 }
{ 128.66 115.91 115.38 138.74 133.69 123.72 92.72 112.07 155.05 131.99 111.71 126.48 125.62 132.33 117.57 134.86 149.28 149.17 140.22 131.33 137.12 115.99 111.58 136.72 152.31 116.51 133.04 101.31 137.74 145.93 140.65 163.61 }
{ 130.82 126.52 112.57 128.29 141.79 120.70 107.73 97.36 130.01 118.06 110.84 127.10 117.07 129.87 133.42 131.36 155.77 139.62 134.76 122.15 137.53 106.49 121.91 119.60 134.19 108.67 138.29 92.90 120.56 123.40 132.56 150.05 }
{ 111.85 125.86 105.50 127.08 131.94 114.84 86.41 92.82 135.16 123.65 103.12 136.64 120.40 109.80 122.60 133.95 145.98 143.53 123.27 109.96 141.23 113.44 120.99 138.88 130.90 107.40 124.30 91.19 139.37 131.61 124.52 142.84 }
```

Colonne :

```
// Add vectors in parallel.
cudaError_t cudaStatus = multiplyWithCuda(c, a, b, arraySize, KERNEL_COLUMN);
```

Matrice A :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }
{ 3.64 1.82 3.69 2.66 0.60 2.55 2.28 1.69 3.77 2.16 2.31 2.15 1.02 1.59 1.40 0.15 3.18 0.79 0.28 1.56 2.36 0.28 0.79 0.62 2.58 1.82 2.42 2.79 1.77 2.74 1.59 3.34 }
{ 2.37 0.80 3.80 3.50 1.57 3.95 0.74 3.58 2.30 1.77 2.51 2.83 0.20 1.14 1.04 1.63 3.58 2.84 2.91 3.58 1.57 1.59 3.62 1.23 1.55 2.28 1.41 2.93 2.98 2.95 2.96 0.56 }
{ 0.80 1.09 2.72 3.64 1.47 2.07 0.44 3.63 0.81 2.07 2.62 1.75 2.75 0.36 0.32 0.30 0.11 1.42 1.23 2.79 0.57 1.58 0.27 2.70 2.90 0.79 2.78 2.46 2.62 2.18 0.39 2.18 }
{ 0.20 3.90 1.86 3.88 2.91 2.12 2.72 2.77 1.49 1.55 0.20 3.24 0.68 0.24 1.24 3.14 0.85 1.29 1.94 0.04 1.76 2.03 2.42 3.03 1.60 1.40 3.37 3.77 2.49 3.99 0.21 2.25 }
{ 3.45 1.55 1.23 1.14 0.11 2.45 1.56 0.07 0.12 1.82 3.43 3.75 1.61 2.91 2.45 3.23 3.50 2.33 2.38 0.32 3.17 1.85 2.36 2.36 2.28 3.26 2.69 1.37 0.04 1.07 1.37 0.93 }
{ 3.08 1.49 3.48 0.47 2.90 2.52 3.58 0.73 2.14 3.61 3.12 3.86 0.79 3.61 3.19 3.81 3.40 3.56 2.51 0.74 2.47 1.06 1.00 2.93 3.15 0.68 2.86 1.92 0.33 3.23 3.42 0.50 }
{ 1.23 2.29 1.89 3.05 0.77 2.91 3.98 0.06 2.59 2.77 0.70 0.29 1.10 2.71 3.36 2.23 1.48 3.12 0.93 2.55 0.93 0.82 3.88 1.13 0.09 3.08 0.61 0.19 0.04 1.39 2.58 1.37 }
{ 1.66 1.67 0.36 2.18 2.80 1.35 2.63 3.50 2.14 2.28 0.22 1.86 1.04 1.71 3.45 1.95 0.35 2.26 2.84 3.11 3.46 3.40 1.17 3.46 2.58 2.99 3.31 3.06 3.49 3.63 3.65 2.09 }
{ 1.76 0.43 1.42 3.29 1.71 0.68 0.00 2.63 3.09 0.25 0.36 1.54 2.74 3.23 0.62 0.57 0.19 1.38 0.07 3.91 2.59 0.21 3.66 2.55 1.73 0.08 0.97 1.57 2.62 3.13 0.09 3.30 }
{ 0.55 2.54 2.20 3.93 1.85 0.86 1.20 3.13 1.37 1.92 2.43 0.41 3.94 3.02 0.60 1.83 2.85 3.99 3.10 2.57 2.48 3.78 2.75 1.16 2.27 1.16 2.19 2.85 0.99 2.19 1.97 3.81 }
{ 2.68 3.06 2.49 1.78 3.20 2.98 3.96 0.82 2.52 3.64 2.37 3.67 2.02 1.96 1.80 3.56 3.76 0.73 1.01 0.81 2.92 3.76 2.50 0.17 2.57 0.14 2.23 2.18 1.79 0.25 3.98 0.51 }
{ 2.40 0.20 0.69 2.64 1.33 1.26 2.61 1.64 1.71 0.89 2.71 0.67 1.99 2.03 3.21 1.74 0.02 1.36 1.50 0.51 3.31 2.93 3.56 2.57 2.60 2.06 2.94 2.95 3.83 1.86 1.45 0.45 }
{ 3.62 3.28 1.82 0.43 3.48 3.05 0.16 1.67 3.74 0.27 0.66 0.85 0.47 2.36 0.47 2.47 2.32 2.76 2.93 3.66 2.68 1.19 3.24 1.51 3.99 3.09 0.82 0.68 1.55 0.72 2.32 0.78 }
{ 3.84 0.76 1.53 1.22 0.48 0.79 0.18 3.08 3.04 3.52 0.01 1.79 1.71 3.21 1.85 1.98 3.00 0.95 2.24 0.37 3.69 3.58 3.43 0.24 1.96 2.53 2.99 3.02 1.02 3.64 3.17 0.80 }
{ 2.21 0.20 2.09 3.16 2.82 0.10 2.93 2.21 3.66 3.50 3.19 1.22 1.58 1.93 0.21 2.61 0.00 3.07 3.02 0.88 0.62 0.23 2.68 0.13 1.38 2.79 3.19 1.75 3.93 2.54 1.18 1.91 }
{ 3.50 0.32 0.40 0.57 0.57 3.67 1.23 1.50 3.26 0.19 3.26 0.96 3.59 1.03 1.08 0.83 0.92 3.38 3.50 3.55 3.38 2.44 0.16 0.69 0.74 1.42 1.80 0.74 3.58 2.49 2.19 3.65 }
```

Matrice B :

```
Matrix B:
{ 2.09 2.92 4.00 3.88 3.47 3.36 0.59 1.50 1.14 0.68 3.90 2.97 2.87 0.12 2.58 3.25 2.94 3.27 0.50 2.88 1.50 2.60 3.66 3.56 0.13 3.09 1.80 0.03 0.91 1.67 0.63 3.68 }
{ 2.01 0.97 3.02 3.63 0.85 0.13 3.08 0.91 2.23 3.23 3.41 1.33 2.16 3.97 3.48 2.59 0.91 3.25 1.48 1.47 0.14 2.25 1.59 0.48 1.30 3.60 0.81 3.76 0.50 3.40 3.58 1.78 }
{ 0.89 2.54 2.24 2.50 1.69 0.07 3.46 0.56 1.07 1.17 2.73 2.25 3.84 2.43 3.15 1.69 3.27 2.96 0.54 2.07 0.08 0.03 2.99 1.34 3.61 1.85 3.07 0.20 2.52 0.10 2.73 3.88 }
{ 2.00 2.83 1.06 3.29 2.59 2.39 1.05 3.38 1.84 2.99 1.41 0.06 1.91 2.51 1.62 1.53 0.11 1.16 2.39 3.54 1.73 3.73 1.53 2.13 0.04 3.90 3.46 1.75 2.80 1.33 2.41 1.99 }
{ 2.24 0.16 1.51 0.93 3.75 3.44 3.86 2.44 2.34 0.31 1.63 1.22 1.99 0.32 3.50 1.22 3.13 3.73 2.35 0.86 1.12 2.24 3.50 1.00 1.21 2.91 3.47 0.92 0.76 1.79 3.56 2.30 }
{ 0.12 3.98 0.24 2.48 1.03 0.68 0.46 0.93 1.98 3.84 0.20 0.32 1.35 1.12 2.06 1.96 1.65 3.70 2.40 1.17 1.90 1.61 3.81 2.19 3.34 3.18 3.21 2.37 3.48 1.27 3.76 1.07 }
{ 3.77 2.88 0.51 3.10 1.89 0.70 0.11 0.17 1.82 3.45 0.54 0.06 0.85 3.27 3.19 3.12 2.63 1.19 3.13 3.69 3.37 0.97 2.00 0.09 0.30 0.12 1.42 2.86 1.64 0.72 1.19 0.73 }
{ 2.98 0.54 0.23 2.10 1.44 3.23 0.64 3.60 0.53 0.12 0.73 3.27 1.00 2.28 1.39 2.16 2.66 3.48 3.73 3.78 0.42 0.61 2.23 3.20 3.58 0.29 3.49 3.29 1.55 2.20 2.69 1.80 }
{ 1.13 3.80 3.93 2.91 2.38 1.09 0.53 0.79 2.40 1.95 1.42 0.92 1.91 2.77 3.79 2.68 3.93 0.95 2.05 0.94 3.59 1.94 1.31 2.15 2.28 1.35 1.39 0.68 1.16 0.88 3.69 2.96 }
{ 3.23 2.60 2.75 1.08 3.07 0.26 2.89 3.44 1.37 0.18 1.94 0.08 3.35 2.47 2.16 2.42 2.27 3.11 2.56 2.79 3.04 3.47 0.63 1.69 3.15 0.09 1.75 0.17 2.03 1.08 0.27 0.98 }
{ 3.93 2.03 2.46 0.28 1.47 3.82 3.44 1.35 0.97 1.16 0.20 3.97 0.69 0.28 0.88 1.20 3.15 2.60 0.33 1.05 0.55 0.87 2.25 0.71 2.53 0.43 1.81 3.04 1.34 0.23 1.13 2.78 }
{ 3.49 2.27 3.47 0.32 3.18 4.00 2.54 2.48 2.38 2.37 0.78 3.92 0.93 0.17 0.96 1.02 1.63 3.18 1.09 3.60 1.52 2.91 0.83 0.21 2.42 1.42 1.44 2.67 3.90 0.89 3.58 1.39 }
{ 3.81 2.66 0.81 1.13 1.42 2.13 1.16 0.88 0.40 0.43 2.84 3.38 2.69 3.28 0.02 1.97 2.42 3.74 1.16 2.42 2.99 3.50 3.65 3.32 2.42 1.22 3.14 0.60 2.17 2.76 1.58 3.97 }
{ 0.41 1.65 0.11 2.75 1.04 0.68 0.91 0.76 3.45 3.05 0.19 1.93 0.22 3.84 0.34 2.26 3.85 2.30 3.51 1.63 1.61 1.65 2.63 3.29 3.78 1.13 0.01 2.47 0.20 2.62 1.69 3.82 }
{ 2.42 0.80 1.66 1.36 2.49 3.23 0.02 0.42 1.58 1.11 1.27 3.30 3.41 1.25 0.79 2.26 1.88 3.98 1.73 1.47 1.68 1.54 2.32 3.51 1.35 1.57 3.80 2.04 1.67 2.43 3.11 0.33 }
{ 0.60 0.83 0.35 1.27 1.89 2.52 1.33 0.32 2.46 0.57 1.12 3.30 1.76 3.98 0.88 0.43 2.08 1.06 0.03 2.31 3.08 0.43 3.88 1.48 3.62 2.09 0.89 1.21 1.91 1.24 1.66 3.06 }
{ 3.37 1.56 1.71 2.90 2.94 0.37 0.33 3.01 3.39 2.55 1.44 1.90 0.38 1.66 0.75 1.37 3.65 0.92 1.24 0.62 1.37 0.73 1.09 0.82 0.69 1.34 0.26 2.01 3.85 3.24 1.79 3.65 }
{ 0.72 1.93 1.89 2.80 0.99 3.75 1.11 1.20 3.29 3.28 2.52 3.18 0.03 1.04 1.16 3.81 3.44 2.07 2.83 1.02 3.68 1.97 0.37 3.30 0.31 1.13 1.54 2.57 2.06 1.73 0.56 3.90 }
{ 0.30 0.02 3.66 1.65 3.02 0.75 0.41 1.69 2.71 2.11 3.30 3.35 1.09 2.00 3.48 0.60 3.88 2.15 0.88 0.27 0.82 0.79 0.77 0.80 3.83 2.37 2.96 1.18 2.72 3.14 3.69 3.71 }
{ 3.38 1.04 0.42 0.25 3.62 0.59 3.57 1.44 3.83 1.53 1.33 2.74 3.55 1.52 1.73 2.70 1.07 1.93 2.11 3.21 3.18 2.99 1.95 3.31 2.24 1.12 1.39 0.22 3.65 3.05 1.47 1.13 }
{ 0.84 1.30 0.20 3.53 1.00 1.62 0.58 3.50 2.43 3.13 0.79 1.81 3.05 1.73 2.55 3.22 0.70 1.29 3.85 2.84 0.79 3.45 0.19 2.97 0.14 0.52 0.33 2.21 2.28 2.13 0.88 0.31 }
{ 1.22 0.26 1.65 0.53 1.01 3.28 2.04 1.07 3.95 3.67 0.49 1.14 2.83 1.88 1.18 1.38 2.10 3.24 2.07 0.27 3.32 2.22 0.41 2.04 3.79 2.89 1.90 1.81 3.68 2.20 1.46 3.54 }
{ 2.64 0.49 2.03 0.45 1.12 2.19 3.17 1.48 3.43 2.98 1.86 1.44 1.77 2.34 1.52 2.53 2.23 1.93 3.35 0.74 0.04 0.44 0.16 0.89 1.02 1.64 2.83 1.77 0.53 2.18 3.43 3.23 }
{ 0.87 3.70 1.25 0.57 2.43 0.04 3.82 2.15 1.55 0.05 0.95 3.54 2.37 0.61 0.07 2.14 3.67 2.61 0.90 1.62 3.86 0.39 1.60 2.02 1.34 1.31 2.50 1.00 1.16 2.13 3.59 1.91 }
{ 1.31 2.70 1.84 2.33 1.65 3.69 0.48 2.66 0.59 0.99 0.45 0.41 2.76 3.31 1.96 1.81 2.61 1.67 2.56 0.40 3.32 1.47 3.51 2.08 3.85 1.58 2.90 2.52 1.50 3.25 3.41 0.26 }
{ 2.97 1.45 0.94 3.28 3.13 1.10 2.25 0.70 0.50 3.03 1.66 1.12 0.68 2.18 0.96 1.27 0.90 2.63 2.79 2.68 2.57 0.22 1.32 3.81 1.28 1.23 0.65 1.47 1.11 3.84 0.94 1.04 }
{ 2.50 3.02 2.33 1.08 2.85 1.35 2.86 0.09 2.80 2.52 0.51 2.88 1.52 2.10 3.46 2.37 1.98 0.12 2.69 2.21 1.40 2.29 3.06 0.09 0.00 2.19 2.99 1.13 3.18 3.34 2.90 1.19 }
{ 2.15 1.62 0.41 1.56 0.39 2.67 0.65 2.76 2.68 2.95 3.55 3.03 2.28 1.06 0.45 1.07 2.93 3.91 1.95 0.86 3.20 2.45 1.39 1.93 3.99 1.49 3.79 0.88 2.52 2.30 0.58 3.72 }
{ 1.83 2.57 0.19 2.62 2.97 0.40 2.60 0.80 0.94 2.93 3.98 3.02 3.57 2.19 3.57 3.48 2.46 2.88 2.92 0.61 3.02 0.61 1.62 2.46 2.87 3.32 3.54 0.25 1.58 2.41 1.95 1.81 }
{ 3.05 2.32 1.34 2.33 1.44 3.00 0.33 0.38 3.99 0.18 1.59 0.43 1.66 1.74 1.08 1.81 0.90 1.05 2.57 3.57 3.52 1.80 1.65 0.93 3.86 2.96 1.37 2.89 2.41 2.49 3.58 2.73 }
{ 0.94 0.44 2.13 2.86 2.27 0.06 2.04 1.83 3.88 3.56 2.81 0.98 0.41 1.07 1.87 2.43 0.20 1.54 0.47 3.99 2.96 0.16 0.02 2.72 3.83 2.83 3.00 0.56 2.71 2.45 2.65 3.90 }
{ 0.28 3.63 3.45 2.51 3.56 2.30 0.61 3.59 2.30 1.05 0.70 2.50 3.34 2.68 3.43 1.79 3.93 2.64 2.84 0.23 3.26 2.99 3.90 3.34 1.26 0.47 0.82 1.17 3.79 3.37 1.00 0.86 }
```

Matrice C :

```
Matrix C:
{ 115.31 112.85 89.07 117.14 114.76 107.13 85.01 103.48 118.60 113.65 86.02 106.86 110.85 114.08 111.23 117.48 121.38 137.25 133.00 116.29 117.33 119.75 100.92 114.01 117.46 94.55 117.04 96.14 121.42 114.85 121.94 115.02 }
{ 145.91 125.33 123.53 123.67 153.73 123.83 122.57 116.21 157.91 145.27 99.26 136.91 128.82 120.27 134.54 138.91 152.49 156.30 134.98 130.46 136.59 128.05 116.70 123.72 136.33 121.41 143.00 109.04 155.78 138.50 157.24 150.29 }
{ 137.50 124.73 101.68 139.33 137.33 118.06 103.31 107.35 143.97 145.36 105.93 141.69 123.67 137.75 128.79 135.89 162.62 168.40 148.39 118.06 138.02 115.35 131.66 133.97 148.53 111.74 139.36 121.85 138.59 151.87 147.46 143.69 }
{ 117.70 135.91 93.46 144.23 126.97 118.97 90.21 110.35 138.83 125.60 105.08 133.36 125.50 136.56 112.10 134.66 156.71 152.45 133.65 126.10 149.87 108.18 136.62 146.27 159.53 116.07 138.83 104.53 141.93 137.78 141.21 162.83 }
{ 124.71 127.36 116.30 125.51 132.06 113.46 102.95 102.64 146.86 121.40 105.57 126.02 132.23 125.80 122.11 127.20 158.39 155.82 125.88 118.01 146.89 110.66 119.78 121.26 163.04 112.01 143.75 103.46 139.16 135.20 157.31 150.81 }
{ 118.91 110.49 97.77 117.59 137.56 102.87 96.66 96.45 114.12 109.36 93.08 114.28 108.44 114.03 111.85 110.64 136.19 130.36 109.50 111.78 122.92 98.25 114.76 108.32 114.95 93.45 108.71 85.62 126.15 114.55 112.97 122.24 }
{ 125.17 116.00 107.08 125.76 134.57 114.03 117.36 92.14 139.17 130.82 117.34 145.06 121.34 119.49 111.22 134.09 154.40 158.62 111.29 116.33 148.12 93.73 115.26 129.92 134.35 116.21 131.48 96.42 129.19 129.93 132.01 162.99 }
{ 139.36 135.76 121.47 128.48 150.65 124.34 128.31 106.99 151.74 129.83 109.31 148.63 132.68 135.28 132.00 146.04 164.68 167.47 139.92 129.94 149.82 118.47 135.30 136.03 154.19 118.00 144.40 115.04 140.66 146.04 162.67 152.16 }
{ 127.27 136.15 109.33 125.07 143.82 113.29 109.35 103.94 132.05 111.94 101.98 116.12 119.25 118.35 118.43 127.01 152.99 139.98 120.58 119.33 142.56 107.70 126.45 113.02 118.87 112.52 129.78 90.66 127.42 117.40 137.23 143.57 }
{ 137.20 131.54 102.76 134.00 140.31 136.68 119.45 113.76 149.81 124.63 104.62 142.12 126.18 135.11 118.36 134.46 161.32 160.91 136.55 136.27 132.92 111.43 150.36 131.94 154.39 130.19 151.38 115.28 138.51 137.53 160.72 170.42 }
{ 128.43 126.64 118.55 132.23 129.24 116.91 104.44 106.77 158.03 132.62 107.33 134.40 117.87 123.04 106.58 134.33 159.06 158.94 125.99 122.37 139.45 105.99 115.59 135.25 151.73 115.87 131.19 109.83 139.15 133.75 146.56 173.39 }
{ 115.78 95.16 98.73 116.85 132.23 104.03 102.51 86.55 138.20 122.25 109.03 124.97 113.30 114.45 106.63 119.29 135.06 152.53 112.44 105.95 129.11 100.41 112.06 126.65 132.61 114.42 118.13 91.29 119.66 135.29 125.85 141.63 }
{ 105.01 119.91 84.80 120.02 109.41 102.45 82.32 88.82 108.67 116.01 79.72 108.74 100.78 112.77 106.51 123.60 143.58 133.41 124.29 102.37 121.27 89.33 107.06 110.97 125.97 85.47 112.18 98.55 111.75 99.98 119.38 120.75 }
{ 126.07 126.63 102.76 134.81 143.06 108.45 113.17 105.82 143.73 127.10 95.33 127.34 129.99 139.44 128.54 134.59 162.04 162.70 143.38 119.40 142.01 116.38 139.23 138.08 144.25 118.75 134.13 112.12 134.05 152.08 156.22 143.54 }
{ 135.73 128.13 93.73 141.27 129.54 122.70 105.87 112.69 142.78 129.07 106.39 129.41 126.93 145.22 122.40 136.55 157.18 159.79 146.78 134.21 145.02 117.77 136.32 131.27 158.38 107.08 132.52 116.15 136.42 139.31 145.38 150.03 }
{ 112.54 114.52 107.22 108.49 138.66 117.31 127.79 104.42 141.61 108.38 101.70 136.71 120.46 115.56 117.58 116.76 143.59 155.63 120.74 130.66 131.10 109.08 128.17 115.99 155.76 105.88 132.94 94.77 139.57 128.37 135.21 145.20 }
{ 127.40 142.08 115.55 142.75 135.85 110.81 91.86 107.76 130.05 122.76 102.00 118.30 127.00 121.88 128.14 132.18 148.32 143.21 128.04 123.46 131.35 114.66 128.27 124.25 128.90 107.94 127.06 98.63 136.87 126.59 134.72 137.24 }
{ 144.17 134.65 118.85 145.74 153.62 130.84 122.08 123.39 165.20 150.52 124.31 147.12 133.55 129.32 134.08 145.93 164.51 169.61 145.24 141.69 144.72 114.51 127.72 141.78 166.10 135.90 162.37 114.53 161.98 145.97 166.77 176.58 }
{ 112.68 116.80 85.11 98.47 115.06 105.81 99.08 98.05 104.26 92.05 85.79 116.21 115.98 103.21 100.24 107.59 125.12 134.67 111.79 105.89 118.74 102.20 114.65 108.35 125.70 96.60 132.72 84.37 121.89 112.22 122.48 116.23 }
{ 127.18 127.20 104.51 128.11 131.95 123.91 108.96 108.42 143.11 128.00 105.19 126.24 127.48 133.17 128.26 124.91 143.43 151.65 138.22 125.04 137.41 114.68 124.34 107.20 137.71 126.96 146.16 114.69 137.89 135.96 159.67 137.95 }
{ 123.85 119.20 110.06 123.90 129.25 121.29 98.30 98.23 136.19 129.67 92.60 137.04 107.31 116.32 100.08 121.31 145.96 143.82 116.07 117.35 123.00 104.03 117.95 119.42 124.69 104.67 115.71 100.21 131.73 131.67 130.85 143.85 }
{ 147.28 148.76 136.43 152.02 160.24 142.36 121.92 117.32 173.51 143.81 117.27 157.23 135.16 142.14 138.40 153.92 187.51 173.72 141.08 149.99 163.27 123.79 148.53 136.77 169.40 128.95 152.90 127.81 158.00 146.88 172.34 180.29 }
{ 110.03 108.20 91.62 121.83 117.11 87.13 89.32 78.00 131.55 125.33 88.40 94.78 101.56 121.26 103.93 123.40 122.88 126.30 119.18 115.16 123.21 93.01 100.49 116.85 106.80 97.18 109.50 92.45 110.41 113.53 121.93 126.82 }
{ 145.58 137.25 117.24 151.22 164.42 140.16 121.38 125.44 172.00 149.30 123.31 152.52 151.94 145.41 144.73 159.75 166.00 177.93 165.71 152.47 177.86 132.26 135.75 159.55 167.38 137.12 168.01 119.86 160.32 171.58 170.72 159.47 }
{ 101.72 110.99 80.06 103.40 111.83 101.80 87.45 94.69 118.57 91.89 83.07 110.70 114.84 107.51 96.20 116.25 124.91 121.04 122.91 102.87 116.34 104.52 106.53 119.35 109.05 90.76 111.92 79.09 105.51 114.14 121.19 119.42 }
{ 138.23 119.17 115 143.39 142.31 137.94 114.57 128.46 167.44 145.45 117.07 147.79 136.77 153.31 129.19 144.37 172.88 167.29 153.16 128.30 153.29 134.11 134.01 148.04 157.91 124.05 150.35 120.48 158.37 160.84 150.07 180.24 }
{ 146.08 132.38 125.47 144.20 148.91 129.28 120.13 117.49 162.78 154.76 114.92 133.85 137.58 144.05 141.60 145.24 163.53 160.50 136.58 138.31 146.26 126.67 135.47 124.99 157.65 131.61 149.41 116.34 154.65 138.23 158.62 168.37 }
{ 122.87 118.15 91.43 120.75 122.71 123.87 98.26 94.05 129.51 127.72 99.24 128.66 124.72 119.73 111.11 131.70 141.20 144.89 134.44 113.60 134.65 103.22 115.04 126.60 128.71 111.46 143.84 99.01 118.25 129.60 134.15 137.53 }
{ 108.37 111.48 111.42 137.49 136.18 110.37 107.36 100.80 143.45 129.83 109.28 123.12 119.39 125.37 130.01 133.69 150.66 146.33 128.26 108.22 127.87 100.24 123.47 133.85 135.72 120.62 125.11 99.06 120.02 142.34 150.84 149.32 }
{ 128.66 115.91 115.38 138.74 133.69 123.72 92.72 112.07 155.05 131.99 111.71 126.48 125.62 132.33 117.57 134.86 149.28 149.17 140.22 131.33 137.12 115.99 111.58 136.72 152.31 116.51 133.04 101.31 137.74 145.93 140.65 163.61 }
{ 130.82 126.52 112.57 128.29 141.79 120.70 107.73 97.36 130.01 118.06 110.84 127.10 117.07 129.87 133.42 131.36 155.77 139.62 134.76 122.15 137.53 106.49 121.91 119.60 134.19 108.67 138.29 92.90 120.56 123.40 132.56 150.05 }
{ 111.85 125.86 105.50 127.08 131.94 114.84 86.41 92.82 135.16 123.65 103.12 136.64 120.40 109.80 122.60 133.95 145.98 143.53 123.27 109.96 141.23 113.44 120.99 138.88 130.90 107.40 124.30 91.19 139.37 131.61 124.52 142.84 }
```

b. Les blocs doivent avoir une taille de 8x8 (à l'exception du cas de matrice 4x4)

B. Profilez votre kernel et discuter des résultats

Avec des matrices 32*32 :

NvProf :

Élément :

```
==21736== Profiling application: a
==21736== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   62.24%  6.7520us         1  6.7520us  6.7520us  6.7520us  multiplyKernelElement(float*, float const *, float const *, int)
                   28.32%  3.0720us         2  1.5360us  1.2800us  1.7920us  [CUDA memcpy HtoD]
                    9.44%  1.0240us         1  1.0240us  1.0240us  1.0240us  [CUDA memcpy DtoH]
      API calls:   79.26%  221.22ms         3  73.740ms  2.2000us  221.21ms  cudaMalloc
                   20.52%  57.277ms         1  57.277ms  57.277ms  57.277ms  cudaDeviceReset
                    0.12%  332.70us         3  110.90us  42.400us  232.90us  cudaMemcpy
                    0.04%  124.60us         3  41.533us  2.3000us  117.40us  cudaFree
                    0.03%  75.500us         1  75.500us  75.500us  75.500us  cudaLaunchKernel
                    0.02%  47.500us         1  47.500us  47.500us  47.500us  cudaDeviceSynchronize
                    0.01%  15.600us       101     154ns     100ns  1.1000us  cuDeviceGetAttribute
                    0.01%  14.100us         3  4.7000us     200ns  13.200us  cuDeviceGetCount
                    0.00%  6.0000us         1  6.0000us  6.0000us  6.0000us  cudaSetDevice
                    0.00%  1.5000us         2     750ns     200ns  1.3000us  cuDeviceGet
                    0.00%  1.1000us         1  1.1000us  1.1000us  1.1000us  cuDeviceGetName
                    0.00%     400ns         1     400ns     400ns     400ns  cuDeviceGetLuid
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceTotalMem
                    0.00%     300ns         1     300ns     300ns     300ns  cudaGetLastError
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceGetUuid

C:\Users\lacommap\Desktop\CudaC\Devoir6_Part1>
```

Rangée :

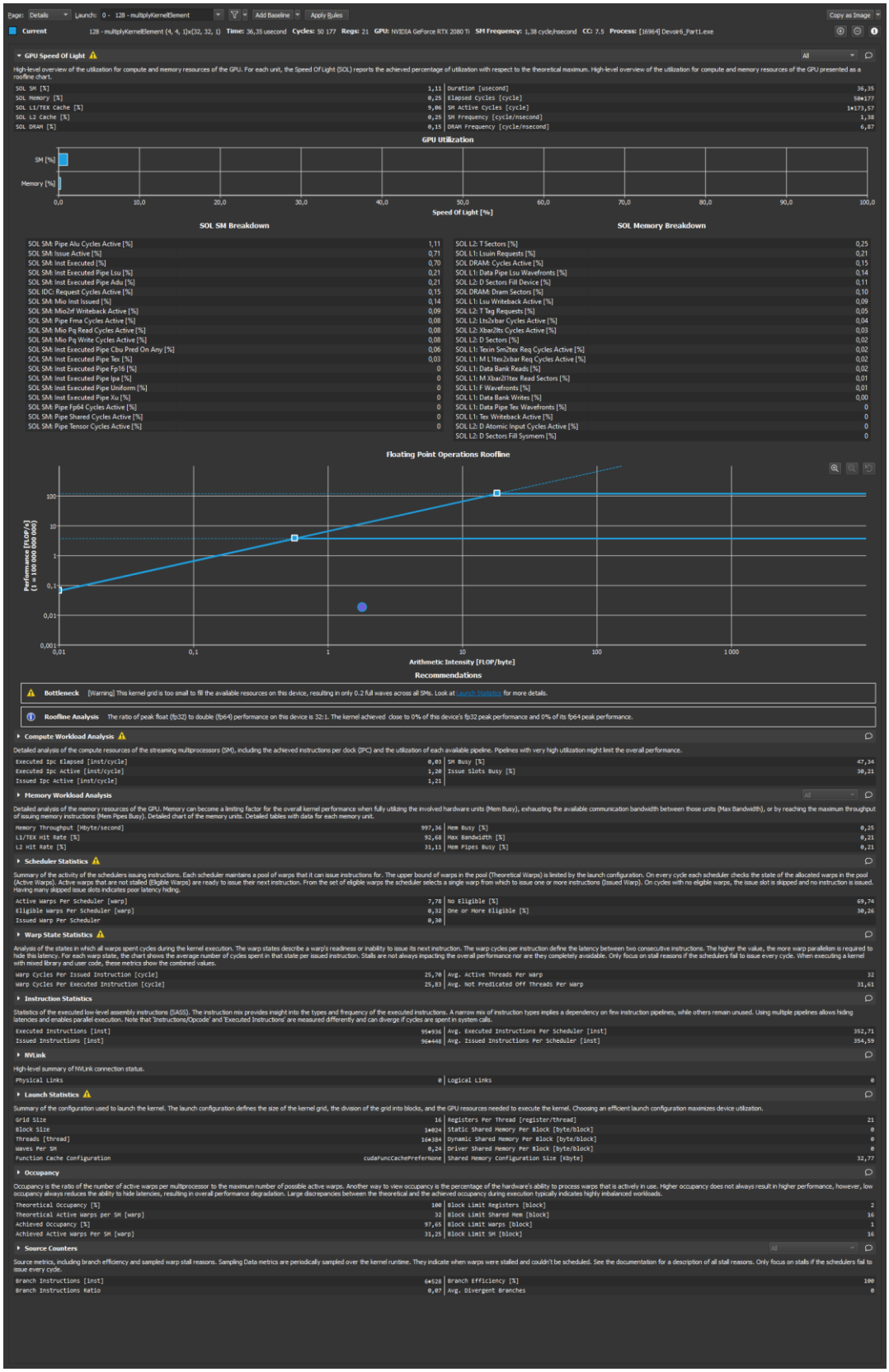
```
==22404== Profiling application: a
==22404== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   96.38%  112.58us         1  112.58us  112.58us  112.58us  multiplyKernelRow(float*, float const *, float const *, int)
                    2.71%  3.1680us         2  1.5840us  1.3120us  1.8560us  [CUDA memcpy HtoD]
                    0.90%  1.0560us         1  1.0560us  1.0560us  1.0560us  [CUDA memcpy DtoH]
      API calls:   78.29%  206.80ms         3  68.933ms  2.0000us  206.79ms  cudaMalloc
                   21.47%  56.707ms         1  56.707ms  56.707ms  56.707ms  cudaDeviceReset
                    0.11%  288.30us         3  96.100us  38.400us  197.00us  cudaMemcpy
                    0.06%  150.40us         1  150.40us  150.40us  150.40us  cudaDeviceSynchronize
                    0.05%  124.30us         3  41.433us  2.0000us  117.10us  cudaFree
                    0.02%  57.200us         1  57.200us  57.200us  57.200us  cudaLaunchKernel
                    0.01%  13.800us       101     136ns     100ns     800ns  cuDeviceGetAttribute
                    0.00%  6.3000us         3  2.1000us     200ns  5.7000us  cuDeviceGetCount
                    0.00%  5.4000us         1  5.4000us  5.4000us  5.4000us  cudaSetDevice
                    0.00%  1.1000us         2     550ns     200ns     900ns  cuDeviceGet
                    0.00%     600ns         1     600ns     600ns     600ns  cuDeviceGetName
                    0.00%     400ns         1     400ns     400ns     400ns  cuDeviceGetLuid
                    0.00%     400ns         1     400ns     400ns     400ns  cudaGetLastError
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceTotalMem
                    0.00%     100ns         1     100ns     100ns     100ns  cuDeviceGetUuid

C:\Users\lacommap\Desktop\CudaC\Devoir6_Part1>
```

Colonne :

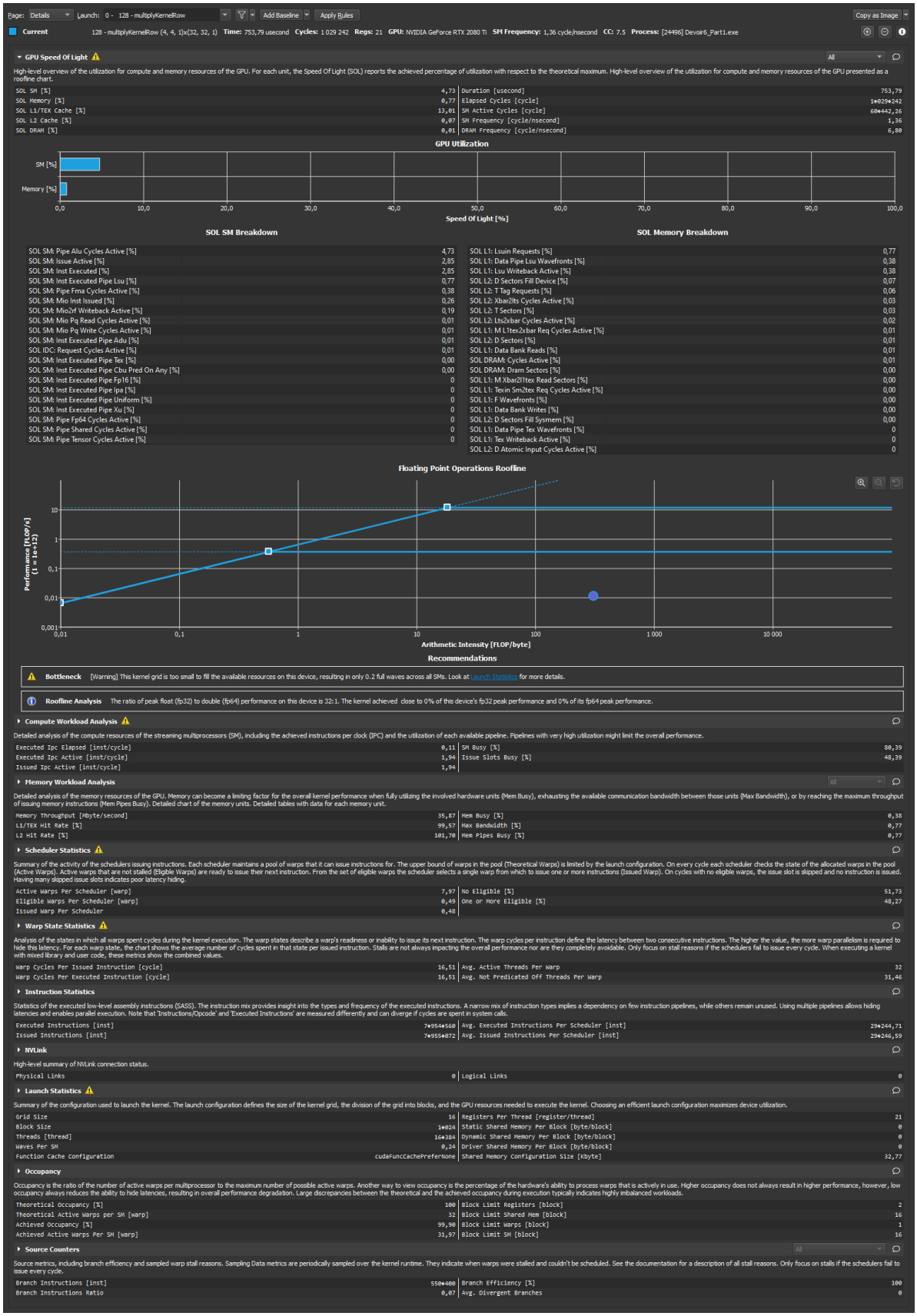
```
==1236== Profiling application: a
==1236== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   96.60%  116.45us         1  116.45us  116.45us  116.45us  multiplyKernelColumn(float*, float const *,
 float const *, int)
                   2.57%  3.1040us         2  1.5520us  1.2800us  1.8240us  [CUDA memcpy HtoD]
                   0.82%     992ns         1     992ns     992ns     992ns  [CUDA memcpy DtoH]
      API calls:   80.31%  226.31ms         3  75.436ms  2.1000us  226.30ms  cudaMalloc
                   19.47%  54.857ms         1  54.857ms  54.857ms  54.857ms  cudaDeviceReset
                    0.10%  274.00us         3  91.333us  37.200us  184.40us  cudaMemcpy
                    0.05%  154.80us         1  154.80us  154.80us  154.80us  cudaDeviceSynchronize
                    0.04%  122.40us         3  40.800us  1.9000us  115.90us  cudaFree
                    0.02%  56.800us         1  56.800us  56.800us  56.800us  cudaLaunchKernel
                    0.01%  14.100us       101     139ns     100ns     600ns  cuDeviceGetAttribute
                    0.00%  6.5000us         3  2.1660us     200ns  5.7000us  cuDeviceGetCount
                    0.00%  5.8000us         1  5.8000us  5.8000us  5.8000us  cudaSetDevice
                    0.00%  1.2000us         2     600ns     200ns  1.0000us  cuDeviceGet
                    0.00%     600ns         1     600ns     600ns     600ns  cuDeviceGetName
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceGetLuid
                    0.00%     300ns         1     300ns     300ns     300ns  cudaGetLastError
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceTotalMem
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceGetUuid

C:\Users\lacommap\Desktop\CudaC\Devoir6_Part1>
```

NSight :

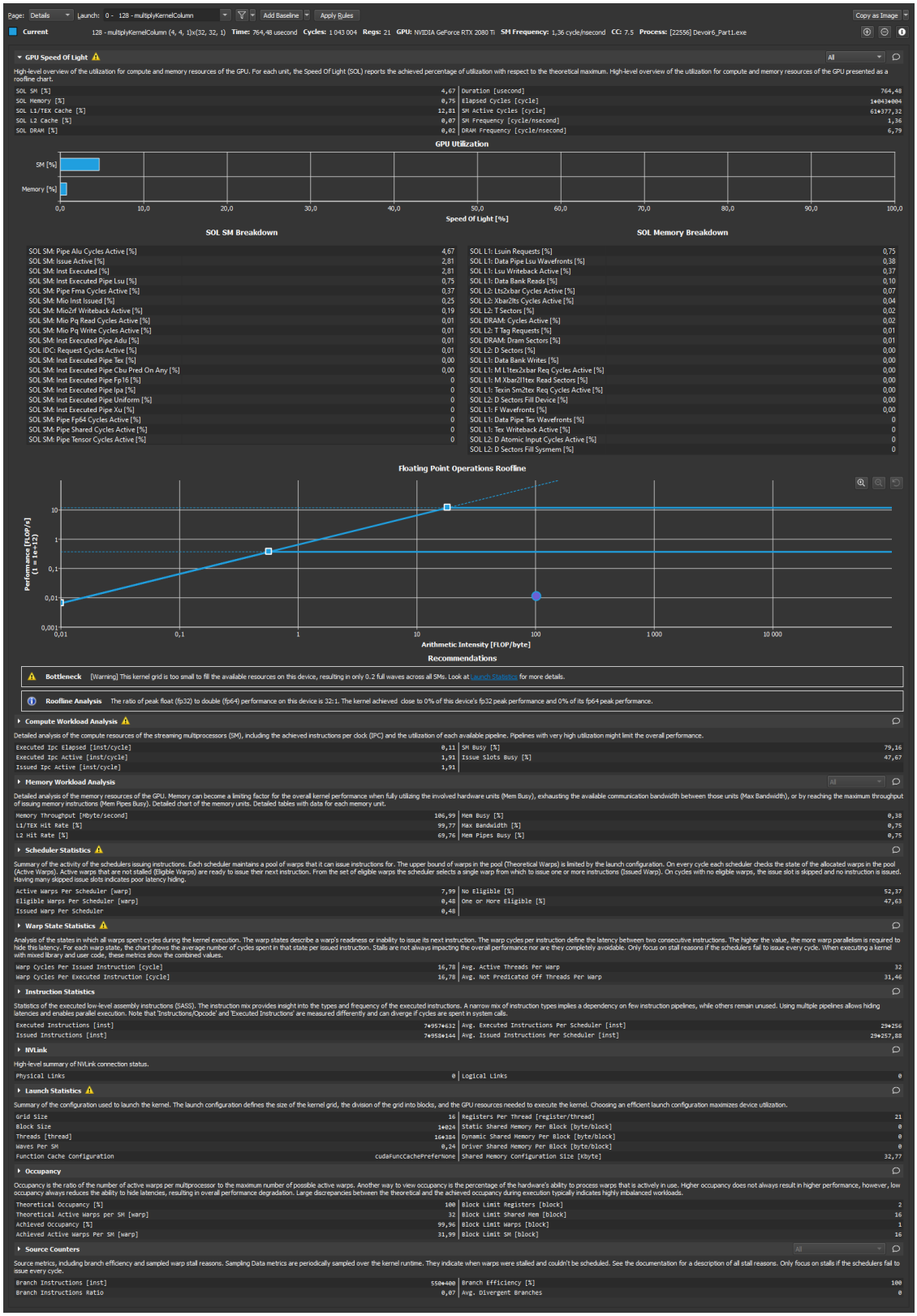
Élément :



Rangée :


Page: Details
Launch: 0 - 128 - multiplyKernelRow
Add Baseline
Apply Rules
Copy as Image
Current
128 - multiplyKernelRow (4, 4, 1)x(32, 32, 1) Time: 753,79 usecond Cycles: 1 029 242 Regs: 21 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,36 cycle/nsecond CC: 7.5 Process: [24496] Devoir6_Part1.exe
GPU Speed Of Light
High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.
SOL SM [%] 4,73
SOL Memory [%] 0,77
SOL L1/TEX Cache [%] 13,01
SOL L2 Cache [%] 0,07
SOL DRAM [%] 0,01
Duration [usecond] 753,79
Elapsed Cycles [cycle] 1♦029♦242
SM Active Cycles [cycle] 68♦442,26
SM Frequency [cycle/nsecond] 1,36
DRAM Frequency [cycle/nsecond] 6,80
GPU Utilization
SM [%]
Memory [%]
0,0 10,0 20,0 30,0 40,0 50,0 60,0 70,0 80,0 90,0 100,0
Speed Of Light [%]
SOL SM Breakdown
SOL SM: Pipe Alu Cycles Active [%] 4,73
SOL SM: Issue Active [%] 2,85
SOL SM: Inst Executed [%] 2,85
SOL SM: Inst Executed Pipe Lsu [%] 0,77
SOL SM: Pipe Fma Cycles Active [%] 0,38
SOL SM: Mio Inst Issued [%] 0,26
SOL SM: Mio2rf Writeback Active [%] 0,19
SOL SM: Mio Pq Read Cycles Active [%] 0,01
SOL SM: Mio Pq Write Cycles Active [%] 0,01
SOL SM: Inst Executed Pipe Adu [%] 0,01
SOL IDC: Request Cycles Active [%] 0,01
SOL SM: Inst Executed Pipe Tex [%] 0,00
SOL SM: Inst Executed Pipe Cbu Pred On Any [%] 0,00
SOL SM: Inst Executed Pipe Fp16 [%] 0
SOL SM: Inst Executed Pipe Ipa [%] 0
SOL SM: Inst Executed Pipe Uniform [%] 0
SOL SM: Inst Executed Pipe Xu [%] 0
SOL SM: Pipe Fp64 Cycles Active [%] 0
SOL SM: Pipe Shared Cycles Active [%] 0
SOL SM: Pipe Tensor Cycles Active [%] 0
SOL Memory Breakdown
SOL L1: Lsuin Requests [%] 0,77
SOL L1: Data Pipe Lsu Wavefronts [%] 0,38
SOL L1: Lsu Writeback Active [%] 0,38
SOL L2: D Sectors Fill Device [%] 0,07
SOL L2: T Tag Requests [%] 0,06
SOL L2: Xbar2lts Cycles Active [%] 0,03
SOL L2: T Sectors [%] 0,03
SOL L2: Lts2xbar Cycles Active [%] 0,02
SOL L1: M L1tex2xbar Req Cycles Active [%] 0,01
SOL L2: D Sectors [%] 0,01
SOL L1: Data Bank Reads [%] 0,01
SOL DRAM: Cycles Active [%] 0,01
SOL DRAM: Dram Sectors [%] 0,00
SOL L1: M Xbar2l1tex Read Sectors [%] 0,00
SOL L1: Texin Sm2tex Req Cycles Active [%] 0,00
SOL L1: F Wavefronts [%] 0,00
SOL L1: Data Bank Writes [%] 0,00
SOL L2: D Sectors Fill Sysmem [%] 0,00
SOL L1: Data Pipe Tex Wavefronts [%] 0
SOL L1: Tex Writeback Active [%] 0
SOL L2: D Atomic Input Cycles Active [%] 0
Floating Point Operations Roofline
Performance [FLOP/s] (1 = 1e+12)
10 1 0,1 0,01 0,001
0,01 0,1 1 10 100 1 000 10 000
Arithmetic Intensity [FLOP/byte]
Recommendations
Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.2 full waves across all SMs. Look at Launch Statistics for more details.
Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.
Compute Workload Analysis
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
Executed Ipc Elapsed [inst/cycle] 0,11
Executed Ipc Active [inst/cycle] 1,94
Issued Ipc Active [inst/cycle] 1,94
SM Busy [%] 80,39
Issue Slots Busy [%] 48,39
Memory Workload Analysis
Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.
Memory Throughput [Mbyte/second] 35,87
L1/TEX Hit Rate [%] 99,57
L2 Hit Rate [%] 101,70
Mem Busy [%] 0,38
Max Bandwidth [%] 0,77
Mem Pipes Busy [%] 0,77
Scheduler Statistics
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.
Active Warps Per Scheduler [warp] 7,97
Eligible Warps Per Scheduler [warp] 0,48
Issued Warp Per Scheduler 0,48
No Eligible [%] 51,73
One or More Eligible [%] 48,27
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.
Warp Cycles Per Issued Instruction [cycle] 16,51
Warp Cycles Per Executed Instruction [cycle] 16,51
Avg. Active Threads Per Warp 32
Avg. Not Predicated Off Threads Per Warp 31,46
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 7♦954♦560
Issued Instructions [inst] 7♦955♦072
Avg. Executed Instructions Per Scheduler [inst] 29♦244,71
Avg. Issued Instructions Per Scheduler [inst] 29♦246,59
NVLink
High-level summary of NVLink connection status.
Physical Links 0
Logical Links 0
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 16
Block Size 1♦024
Threads [thread] 16♦384
Waves Per SM 0,24
Function Cache Configuration cudaFuncCachePreferNone
Registers Per Thread [register/thread] 21
Static Shared Memory Per Block [byte/block] 0
Dynamic Shared Memory Per Block [byte/block] 0
Driver Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [Kbyte] 32,77
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp] 32
Achieved Occupancy [%] 99,90
Achieved Active Warps Per SM [warp] 31,97
Block Limit Registers [block] 2
Block Limit Shared Mem [block] 16
Block Limit Warps [block] 1
Block Limit SM [block] 16
Source Counters
Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.
Branch Instructions [inst] 550♦400
Branch Instructions Ratio 0,07
Branch Efficiency [%] 100
Avg. Divergent Branches 0

Colonne :


Page: Details
Launch: 0 - 128 - multiplyKernelColumn
Add Baseline
Apply Rules
Copy as Image
Current
128 - multiplyKernelColumn (4, 4, 1)x(32, 32, 1) Time: 764,48 usecond Cycles: 1 043 004 Regs: 21 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,36 cycle/nsecond CC: 7.5 Process: [22556] Devoir6_Part1.exe
GPU Speed Of Light
High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.
SOL SM [%] 4,67
SOL Memory [%] 0,75
SOL L1/TEX Cache [%] 12,81
SOL L2 Cache [%] 0,07
SOL DRAM [%] 0,02
Duration [usecond] 764,48
Elapsed Cycles [cycle] 1404344004
SM Active Cycles [cycle] 614377,32
SM Frequency [cycle/nsecond] 1,36
DRAM Frequency [cycle/nsecond] 6,79
GPU Utilization
SM [%]
Memory [%]
0,0 10,0 20,0 30,0 40,0 50,0 60,0 70,0 80,0 90,0 100,0
Speed Of Light [%]
SOL SM Breakdown
SOL SM: Pipe Alu Cycles Active [%] 4,67
SOL SM: Issue Active [%] 2,81
SOL SM: Inst Executed [%] 2,81
SOL SM: Inst Executed Pipe Lsu [%] 0,75
SOL SM: Pipe Fma Cycles Active [%] 0,37
SOL SM: Mio Inst Issued [%] 0,25
SOL SM: Mio2rf Writeback Active [%] 0,19
SOL SM: Mio Pq Read Cycles Active [%] 0,01
SOL SM: Mio Pq Write Cycles Active [%] 0,01
SOL SM: Inst Executed Pipe Adu [%] 0,01
SOL IDC: Request Cycles Active [%] 0,01
SOL SM: Inst Executed Pipe Tex [%] 0,00
SOL SM: Inst Executed Pipe Cbu Pred On Any [%] 0,00
SOL SM: Inst Executed Pipe Fp16 [%] 0
SOL SM: Inst Executed Pipe Ipa [%] 0
SOL SM: Inst Executed Pipe Uniform [%] 0
SOL SM: Inst Executed Pipe Xu [%] 0
SOL SM: Pipe Fp64 Cycles Active [%] 0
SOL SM: Pipe Shared Cycles Active [%] 0
SOL SM: Pipe Tensor Cycles Active [%] 0
SOL Memory Breakdown
SOL L1: Lsuin Requests [%] 0,75
SOL L1: Data Pipe Lsu Wavefronts [%] 0,38
SOL L1: Lsu Writeback Active [%] 0,37
SOL L1: Data Bank Reads [%] 0,10
SOL L2: Lts2xbar Cycles Active [%] 0,07
SOL L2: Xbar2lts Cycles Active [%] 0,04
SOL L2: T Sectors [%] 0,02
SOL DRAM: Cycles Active [%] 0,02
SOL L2: T Tag Requests [%] 0,01
SOL DRAM: Dram Sectors [%] 0,01
SOL L2: D Sectors [%] 0,00
SOL L1: Data Bank Writes [%] 0,00
SOL L1: M L1tex2xbar Req Cycles Active [%] 0,00
SOL L1: M Xbar2l1tex Read Sectors [%] 0,00
SOL L1: Texin Sm2tex Req Cycles Active [%] 0,00
SOL L2: D Sectors Fill Device [%] 0,00
SOL L1: F Wavefronts [%] 0,00
SOL L1: Data Pipe Tex Wavefronts [%] 0
SOL L1: Tex Writeback Active [%] 0
SOL L2: D Atomic Input Cycles Active [%] 0
SOL L2: D Sectors Fill Sysmem [%] 0
Floating Point Operations Roofline
Performance [FLOP/s] (1 e 1e+12)
10 1 0,1 0,01 0,001
0,01 0,1 1 10 100 1 000 10 000
Arithmetic Intensity [FLOP/byte]
Recommendations
Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.2 full waves across all SMs. Look at Launch Statistics for more details.
Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.
Compute Workload Analysis
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
Executed Ipc Elapsed [inst/cycle] 0,11
Executed Ipc Active [inst/cycle] 1,91
Issued Ipc Active [inst/cycle] 1,91
SM Busy [%] 79,16
Issue Slots Busy [%] 47,67
Memory Workload Analysis
Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.
Memory Throughput [Mbyte/second] 106,09
L1/TEX Hit Rate [%] 99,77
L2 Hit Rate [%] 69,76
Mem Busy [%] 0,38
Max Bandwidth [%] 0,75
Mem Pipes Busy [%] 0,75
Scheduler Statistics
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.
Active Warps Per Scheduler [warp] 7,99
Eligible Warps Per Scheduler [warp] 0,48
Issued Warp Per Scheduler 0,48
No Eligible [%] 52,37
One or More Eligible [%] 47,63
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.
Warp Cycles Per Issued Instruction [cycle] 16,78
Warp Cycles Per Executed Instruction [cycle] 16,78
Avg. Active Threads Per Warp 32
Avg. Not Predicated Off Threads Per Warp 31,46
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 7e957e632
Issued Instructions [inst] 7e958e144
Avg. Executed Instructions Per Scheduler [inst] 29e256
Avg. Issued Instructions Per Scheduler [inst] 29e257,88
NVLink
High-level summary of NVLink connection status.
Physical Links 0
Logical Links 0
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 16
Block Size 1e024
Threads [thread] 16e384
Waves Per SM 0,24
Function Cache Configuration cudaFuncCachePreferNone
Registers Per Thread [register/thread] 21
Static Shared Memory Per Block [byte/block] 0
Dynamic Shared Memory Per Block [byte/block] 0
Driver Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [kbyte] 32,77
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp] 32
Achieved Occupancy [%] 99,96
Achieved Active Warps Per SM [warp] 31,99
Block Limit Registers [block] 2
Block Limit Shared Mem [block] 16
Block Limit Warps [block] 1
Block Limit SM [block] 16
Source Counters
Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.
Branch Instructions [inst] 550e400
Branch Instructions Ratio 0,07
Branch Efficiency [%] 100
Avg. Divergent Branches 0

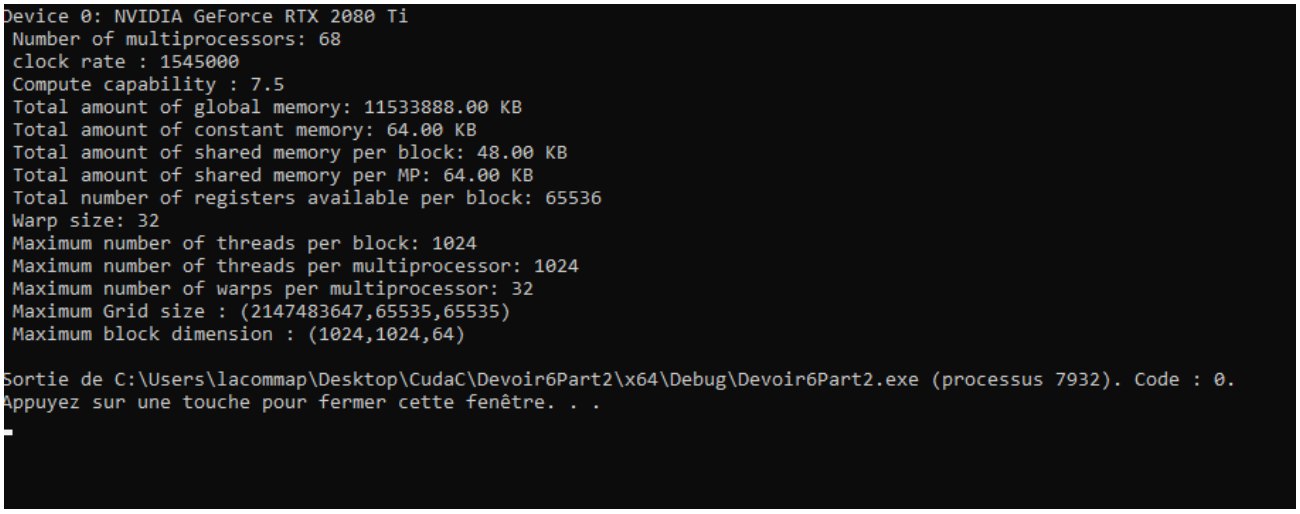
PARTIE 2 – MULTIPLICATION MATRICIELLE OPTIMISÉE

Objectifs

Réimplémentez la multiplication matricielle de la PARTIE 1 en utilisant l'algorithme de tiling [chap. 4, 2]. En programmation paralèlle, le tiling force plusieurs threads à se concentrer sur un sous ensemble de données d'entrée lors de chaque phase de sorte que ce sous ensemble de donnée puisse être stocké dans une mémoire spéciale, la shared memory par exemple, ce qui accélère grandement les accès mémoire. L'étudiant doit démontrer qu'avec des tiles de grande taille, les accès à la mémoire globale n'est plus un principal facteur limitant.
D'autre part, la capacité de raisonner en fonction des limitations matérielles en développant une application est un aspect clé de la pensée computationnelle. Les programmeurs CUDA doivent être conscient des limitations de chaque type de mémoire ainsi que des ressources propres à leur GPU puisqu'elles sont étroitement liées à l'implémentation. Une fois les capacités excédées, le GPU limite le nombre de threads pouvant être exécuter dans chaque SM. Les threads sont assignés aux SM sur une base de bloc par bloc. Chaque device CUDA dispose d'une limitation différente par rapport aux ressources disponible dans les SM, autant une limite de blocs par SM et une limite de threads par SM. La première limitation atteinte deviendra le facteur limitant.
Pour chaque Kernel, l'une ou plusieurs de ces limitations peut devenir le facteur limitant pour le nombre de threads qui sont utilisés simultanément dans le device CUDA.

MÉTHODOLOGIE

1. Votre implémentation doit être « transparent scalable » (voir dection 3.4 à 3.6 dans [chap.3, 2]). Pour ce faire, vous devez connaître les propriétés de votre device. L'annexe B fournit un exemple d'une « host-stub function » qui permet d'extraire les propriétés.



2. **L'implémentation doit être limitée à des matrices carrées 64x64. Toutefois, la taille des tuiles doit varier entre 2x2, 4x4, 8x8 et 16x16.**
3. Validez chaque implementation.

Pour alléger l'affichage je prendrais des matrices 16*16.

Dans la fonction host stub, le nombre de threads par block est équivalent à la taille des tuiles. Par la suite, le nombre de bloc dans un grid est calculé en fonction du nombre de tuilles :

```
// Helper function for using CUDA to add vectors in parallel.
cudaError_t multiplyWithCuda(float* c, const float* a, const float* b, unsigned int size)
{
    float* dev_a = 0;
    float* dev_b = 0;
    float* dev_c = 0;
    cudaError_t cudaStatus;

    if (size >64) {
        printf("Taille de la matrice limité à 64 par 64");
        return cudaErrorInvalidValue;
    }

    // Choose which GPU to run on, change this on a multi-GPU system.
    cudaStatus = cudaSetDevice(0);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaSetDevice failed!  Do you have a CUDA-capable GPU installed?");
        goto Error;
    }

    // Allocate GPU buffers for three vectors (two input, one output)    .
    cudaStatus = cudaMalloc((void**)&dev_c, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }

    cudaStatus = cudaMalloc((void**)&dev_a, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }

    cudaStatus = cudaMalloc((void**)&dev_b, (size * size) * sizeof(float));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed!");
        goto Error;
    }
```

```
// Copy input vectors from host memory to GPU buffers.
cudaStatus = cudaMemcpy(dev_a, a, (size * size) * sizeof(float), cudaMemcpyHostToDevice);
if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaMemcpy failed!");
    goto Error;
}


cudaStatus = cudaMemcpy(dev_b, b, (size * size) * sizeof(float), cudaMemcpyHostToDevice);
if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaMemcpy failed!");
    goto Error;
}



dim3 threadsParBlock(tileLength, tileLength, 1);
dim3 nombreDeBlock((size + tileLength - 1) / tileLength, (size + tileLength - 1) / tileLength);

multiplyKernel << <nombreDeBlock, threadsParBlock >> > (dev_c, dev_a, dev_b, size);


// Check for any errors launching the kernel
cudaStatus = cudaGetLastError();
if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "addKernel launch failed: %s\n", cudaGetErrorString(cudaStatus));
    goto Error;
}

// cudaDeviceSynchronize waits for the kernel to finish, and returns
// any errors encountered during the launch.
cudaStatus = cudaDeviceSynchronize();
if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaDeviceSynchronize returned error code %d after launching addKernel!\n", cudaStatus);
    goto Error;
}

    // Copy output vector from GPU buffer to host memory.
    cudaStatus = cudaMemcpy(c, dev_c, (size * size) * sizeof(float), cudaMemcpyDeviceToHost);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed!");
        goto Error;
    }

Error:
    cudaFree(dev_c);
    cudaFree(dev_a);
    cudaFree(dev_b);

    return cudaStatus;
}
```

Pour le kernel, il existe plusieurs implémentation open-source qui fait ce genre de calcul matricielle, je me suis alors inspiré de ce répertoire sur Github :
https://github.com/tgautam03/CUDA-C/blob/master/05_tiled_mat_mul/tiled_mat_mul_gpu.cu

```
__global__ void multiplyKernel(float* c, const float* a, const float* b, int length) {
    int threadY = threadIdx.y;
    int threadX = threadIdx.x;

    int row = (tileLength * blockIdx.y) + threadY;
    int column = (tileLength * blockIdx.x) + threadX;

    __shared__ float tileA[tileLength][tileLength];
    __shared__ float tileB[tileLength][tileLength];

    float sum = 0;

    int numTiles = (length + tileLength - 1) / tileLength;

    // Data loading
    for (int t = 0; t < numTiles; t++) {
        if (row < length && (t * tileLength + threadX) < length) {
            tileA[threadY][threadX] = a[row * length + (t * tileLength + threadX)];
        }
        else {
            tileA[threadY][threadX] = 0.0f;
        }

        if (column < length && (t * tileLength + threadY) < length) {
            tileB[threadY][threadX] = b[(t * tileLength + threadY) * length + column];
        }
        else {
            tileB[threadY][threadX] = 0.0f;
        }

        __syncthreads();

        for (int k = 0; k < tileLength; k++) {
            sum += tileA[threadY][k] * tileB[k][threadX];
        }

        __syncthreads();
    }

    if (row < length && column < length) {
        c[(row * length) + column] = sum;
    }
}
```

Essentiellement, cette technique permet d'exploiter la mémoire partagée de chaque bloc, ce qui réduit considérablement les accès à la mémoire globale et améliore les performances. Les variables « row » et « column » servent à calculer les indices de la matrice C finale que chaque thread est chargé de calculer.

Dans chaque itération, les données nécessaires sont d'abord exportées depuis la mémoire globale vers la mémoire partagée pour chaque tuile. Une fois toutes les données chargées, on s'assure de leur disponibilité grâce à la fonction __syncthreads(), qui synchronise tous les threads du bloc. Ensuite, les threads effectuent les calculs sur ces données en mémoire partagée. Avant de passer à la tuile suivante, on utilise à nouveau __syncthreads() pour garantir que tous les threads ont terminé leurs calculs.

Finalement, les résultats calculés sont écrits dans la mémoire globale pour remplir les éléments correspondants de la matrice C.

Tuile 2*2 :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 }
{ 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 }
{ 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 }
{ 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 }
{ 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 }
{ 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 }
{ 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 }
{ 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 }
{ 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }

Matrix B:
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 }
{ 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 }
{ 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 }
{ 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 }
{ 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 }
{ 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 }
{ 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 }
{ 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 }
{ 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }

Matrix C:
{ 70.80 70.44 60.55 55.63 53.52 71.79 58.92 74.93 70.70 69.21 45.94 61.33 58.39 76.84 39.43 51.63 }
{ 49.17 55.72 39.75 45.64 41.92 50.29 33.63 54.18 54.02 49.74 42.96 47.97 38.99 63.23 32.08 42.80 }
{ 75.95 76.97 64.24 58.84 65.36 87.00 65.14 83.46 76.66 76.39 53.56 62.77 57.28 88.26 47.16 62.48 }
{ 81.42 72.16 62.48 61.42 69.52 80.93 66.88 74.54 61.90 67.46 55.16 64.62 55.39 89.55 51.41 72.63 }
{ 77.62 83.65 58.76 56.78 68.43 71.75 58.12 76.98 66.41 77.38 51.61 68.67 54.89 88.44 48.11 60.74 }
{ 71.03 68.16 63.49 70.94 66.00 76.27 55.70 78.00 72.34 68.76 58.16 63.07 59.81 82.01 35.69 65.14 }
{ 65.69 67.09 63.95 69.06 62.88 71.89 68.78 67.00 56.89 69.80 46.87 77.88 55.18 88.18 47.56 66.43 }
{ 71.27 68.53 68.56 63.22 66.02 72.17 58.20 72.37 73.04 83.53 55.71 76.59 52.55 79.42 37.49 64.46 }
{ 65.70 70.98 62.60 48.89 67.59 73.68 51.46 77.55 62.28 64.62 53.95 66.86 54.37 81.99 35.07 60.77 }
{ 81.04 68.29 71.31 74.53 71.75 80.50 55.19 80.01 72.40 75.56 58.60 71.48 54.67 86.42 39.28 60.14 }
{ 70.25 64.43 50.70 67.98 61.70 68.88 41.78 63.04 61.80 61.66 58.96 55.29 53.30 74.84 39.85 63.85 }
{ 56.59 49.83 47.54 48.05 48.80 53.69 56.22 58.36 59.48 50.12 43.27 52.23 49.73 61.82 24.73 50.78 }
{ 63.07 65.14 60.88 64.05 63.59 68.71 50.39 67.48 56.20 74.25 57.66 64.72 49.31 77.56 38.70 69.60 }
{ 74.40 75.85 63.54 58.34 60.90 70.58 62.65 70.61 52.94 80.12 45.19 66.55 56.93 70.47 41.53 70.33 }
{ 77.63 83.83 66.00 64.84 72.50 80.28 59.14 85.22 76.03 79.55 57.21 69.55 60.86 90.57 43.11 64.04 }
{ 67.19 74.27 59.60 60.77 65.49 64.51 62.49 72.07 63.26 70.87 56.01 72.25 59.15 87.55 44.04 64.86 }
```

Tuile 4*4 :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 }
{ 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 }
{ 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 }
{ 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 }
{ 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 }
{ 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 }
{ 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 }
{ 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 }
{ 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }

Matrix B:
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 }
{ 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 }
{ 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 }
{ 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 }
{ 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 }
{ 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 }
{ 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 }
{ 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 }
{ 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }

Matrix C:
{ 70.80 70.44 60.55 55.63 53.52 71.79 58.92 74.93 70.70 69.21 45.94 61.33 58.39 76.84 39.43 51.63 }
{ 49.17 55.72 39.75 45.64 41.92 50.29 33.63 54.18 54.02 49.74 42.96 47.97 38.99 63.23 32.08 42.80 }
{ 75.95 76.97 64.24 58.84 65.36 87.00 65.14 83.46 76.66 76.39 53.56 62.77 57.28 88.26 47.16 62.48 }
{ 81.42 72.16 62.48 61.42 69.52 80.93 66.88 74.54 61.90 67.46 55.16 64.62 55.39 89.55 51.41 72.63 }
{ 77.62 83.65 58.76 56.78 68.43 71.75 58.12 76.98 66.41 77.38 51.61 68.67 54.89 88.44 48.11 60.74 }
{ 71.03 68.16 63.49 70.94 66.00 76.27 55.70 78.00 72.34 68.76 58.16 63.07 59.81 82.01 35.69 65.14 }
{ 65.69 67.09 63.95 69.06 62.88 71.89 68.78 67.00 56.89 69.80 46.87 77.88 55.18 88.18 47.56 66.43 }
{ 71.27 68.53 68.56 63.22 66.02 72.17 58.20 72.37 73.04 83.53 55.71 76.59 52.55 79.42 37.49 64.46 }
{ 65.70 70.98 62.60 48.89 67.59 73.68 51.46 77.55 62.28 64.62 53.95 66.86 54.37 81.99 35.07 60.77 }
{ 81.04 68.29 71.31 74.53 71.75 80.50 55.19 80.01 72.40 75.56 58.60 71.48 54.67 86.42 39.28 60.14 }
{ 70.25 64.43 50.70 67.98 61.70 68.88 41.78 63.04 61.80 61.66 58.96 55.29 53.30 74.84 39.85 63.85 }
{ 56.59 49.83 47.54 48.05 48.80 53.69 56.22 58.36 59.48 50.12 43.27 52.23 49.73 61.82 24.73 50.78 }
{ 63.07 65.14 60.88 64.05 63.59 68.71 50.39 67.48 56.20 74.25 57.66 64.72 49.31 77.56 38.70 69.60 }
{ 74.40 75.85 63.54 58.34 60.90 70.58 62.65 70.61 52.94 80.12 45.19 66.55 56.93 70.47 41.53 70.33 }
{ 77.63 83.83 66.00 64.84 72.50 80.28 59.14 85.22 76.03 79.55 57.21 69.55 60.86 90.57 43.11 64.04 }
{ 67.19 74.27 59.60 60.77 65.49 64.51 62.49 72.07 63.26 70.87 56.01 72.25 59.15 87.55 44.04 64.86 }
```

Tuile 8*8 :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 }
{ 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 }
{ 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 }
{ 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 }
{ 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 }
{ 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 }
{ 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 }
{ 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 }
{ 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }

Matrix B:
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 }
{ 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 }
{ 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 }
{ 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 }
{ 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 }
{ 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 }
{ 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 }
{ 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 }
{ 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }

Matrix C:
{ 70.80 70.44 60.55 55.63 53.52 71.79 58.92 74.93 70.70 69.21 45.94 61.33 58.39 76.84 39.43 51.63 }
{ 49.17 55.72 39.75 45.64 41.92 50.29 33.63 54.18 54.02 49.74 42.96 47.97 38.99 63.23 32.08 42.80 }
{ 75.95 76.97 64.24 58.84 65.36 87.00 65.14 83.46 76.66 76.39 53.56 62.77 57.28 88.26 47.16 62.48 }
{ 81.42 72.16 62.48 61.42 69.52 80.93 66.88 74.54 61.90 67.46 55.16 64.62 55.39 89.55 51.41 72.63 }
{ 77.62 83.65 58.76 56.78 68.43 71.75 58.12 76.98 66.41 77.38 51.61 68.67 54.89 88.44 48.11 60.74 }
{ 71.03 68.16 63.49 70.94 66.00 76.27 55.70 78.00 72.34 68.76 58.16 63.07 59.81 82.01 35.69 65.14 }
{ 65.69 67.09 63.95 69.06 62.88 71.89 68.78 67.00 56.89 69.80 46.87 77.88 55.18 88.18 47.56 66.43 }
{ 71.27 68.53 68.56 63.22 66.02 72.17 58.20 72.37 73.04 83.53 55.71 76.59 52.55 79.42 37.49 64.46 }
{ 65.70 70.98 62.60 48.89 67.59 73.68 51.46 77.55 62.28 64.62 53.95 66.86 54.37 81.99 35.07 60.77 }
{ 81.04 68.29 71.31 74.53 71.75 80.50 55.19 80.01 72.40 75.56 58.60 71.48 54.67 86.42 39.28 60.14 }
{ 70.25 64.43 50.70 67.98 61.70 68.88 41.78 63.04 61.80 61.66 58.96 55.29 53.30 74.84 39.85 63.85 }
{ 56.59 49.83 47.54 48.05 48.80 53.69 56.22 58.36 59.48 50.12 43.27 52.23 49.73 61.82 24.73 50.78 }
{ 63.07 65.14 60.88 64.05 63.59 68.71 50.39 67.48 56.20 74.25 57.66 64.72 49.31 77.56 38.70 69.60 }
{ 74.40 75.85 63.54 58.34 60.90 70.58 62.65 70.61 52.94 80.12 45.19 66.55 56.93 70.47 41.53 70.33 }
{ 77.63 83.83 66.00 64.84 72.50 80.28 59.14 85.22 76.03 79.55 57.21 69.55 60.86 90.57 43.11 64.04 }
{ 67.19 74.27 59.60 60.77 65.49 64.51 62.49 72.07 63.26 70.87 56.01 72.25 59.15 87.55 44.04 64.86 }
```

Tuile 16*16 :

```
Matrix A:
{ 0.01 2.25 0.77 3.23 2.34 1.92 1.40 3.58 3.29 2.99 0.70 3.44 2.84 2.05 1.22 0.06 }
{ 0.37 1.46 0.59 0.66 3.95 1.78 0.48 0.02 0.04 1.51 2.13 2.28 2.41 2.43 0.66 2.65 }
{ 1.80 1.41 0.23 2.43 3.13 3.21 2.08 1.21 3.50 2.91 3.82 3.70 2.16 0.57 1.85 0.94 }
{ 3.45 0.84 3.12 3.37 3.99 4.00 2.45 1.57 1.06 1.19 3.36 0.09 1.50 0.37 2.71 0.22 }
{ 0.04 3.68 1.10 1.09 2.35 2.76 3.35 2.91 1.94 0.82 2.97 1.87 1.83 3.80 2.98 0.43 }
{ 2.40 1.54 2.94 2.44 2.29 1.45 0.61 0.90 1.70 3.21 2.07 3.96 3.01 1.38 0.68 2.63 }
{ 1.97 0.25 2.80 2.02 0.59 3.80 0.57 3.62 2.77 1.21 1.71 0.28 3.87 2.73 0.61 3.51 }
{ 3.29 2.33 0.77 0.71 3.27 1.90 0.62 2.02 2.93 1.62 1.12 2.27 2.73 3.02 2.89 1.90 }
{ 0.49 1.47 3.34 0.14 2.07 2.65 1.70 0.42 3.80 3.69 2.20 1.38 1.89 1.50 3.39 1.27 }
{ 1.82 1.09 3.93 1.19 2.96 2.27 0.78 3.05 3.36 1.59 2.00 3.56 0.11 3.98 2.29 0.20 }
{ 2.13 0.78 3.37 2.51 2.63 0.79 3.37 0.49 0.44 2.97 1.26 3.76 1.14 1.35 0.56 2.93 }
{ 3.34 2.83 2.40 2.99 1.01 0.58 0.01 0.24 3.22 3.41 0.84 0.46 2.21 0.06 0.46 1.82 }
{ 3.01 2.74 2.17 0.30 1.75 0.81 2.78 1.16 1.75 0.93 2.31 2.13 2.51 0.64 2.02 3.85 }
{ 2.78 3.70 0.76 1.34 0.71 3.98 1.83 3.99 0.39 2.50 0.38 1.75 3.73 0.19 3.58 1.16 }
{ 0.91 3.08 1.64 0.81 2.51 2.42 1.81 1.87 2.39 2.54 3.42 3.32 2.50 2.88 2.26 1.50 }
{ 0.74 2.95 2.22 3.62 0.97 0.76 2.42 2.79 2.34 1.41 1.98 0.32 2.96 2.45 2.48 2.76 }

Matrix B:
{ 3.22 0.60 2.30 3.47 3.65 2.46 2.91 0.17 2.67 3.91 1.26 2.28 1.22 0.70 0.43 3.48 }
{ 3.40 2.98 0.62 1.31 0.32 0.31 2.56 3.28 2.18 1.79 1.64 1.19 1.86 2.00 0.61 1.29 }
{ 2.95 1.26 3.31 3.84 3.49 2.90 1.20 3.78 0.51 0.26 3.14 2.10 2.44 3.82 0.29 3.50 }
{ 2.62 1.29 0.42 2.02 0.91 1.16 3.68 2.20 2.65 0.46 1.97 1.52 1.99 3.17 2.04 1.53 }
{ 2.75 2.13 2.43 1.58 0.02 2.83 0.40 2.49 3.45 1.97 2.99 1.99 1.52 3.14 2.21 1.43 }
{ 3.82 2.52 0.71 1.50 0.53 2.97 3.81 2.45 0.11 1.32 0.22 2.56 0.53 3.39 3.46 2.39 }
{ 2.89 3.42 0.06 0.51 2.83 2.47 0.87 0.26 0.68 2.50 1.36 1.28 1.47 2.64 3.21 3.23 }
{ 2.11 2.44 3.19 3.60 0.58 2.52 1.61 1.01 0.55 3.42 0.26 1.71 2.29 1.21 2.19 0.90 }
{ 1.25 0.44 3.23 0.54 1.14 3.15 3.58 3.16 2.98 2.46 1.44 3.43 0.91 3.45 0.92 1.00 }
{ 2.17 3.94 0.22 0.33 2.10 1.71 0.38 1.04 3.57 0.93 0.59 0.50 3.73 0.32 0.19 0.23 }
{ 1.35 3.66 1.59 1.73 3.78 3.35 2.14 3.37 2.77 1.59 1.04 0.02 2.10 3.82 1.59 0.96 }
{ 2.34 1.02 2.74 3.78 1.74 3.56 0.03 3.76 2.41 3.14 2.31 0.57 0.89 1.53 0.02 1.67 }
{ 0.33 2.64 3.42 0.26 3.24 2.65 2.77 3.21 2.12 2.74 0.57 2.76 2.91 3.11 0.12 3.47 }
{ 2.58 2.83 0.34 2.21 3.79 0.24 1.10 0.58 3.93 2.48 1.17 3.69 1.47 2.78 0.87 0.62 }
{ 0.96 2.09 3.61 0.43 3.61 1.77 0.32 3.13 0.69 3.90 3.10 3.48 0.84 1.83 0.02 3.00 }
{ 0.46 1.62 1.24 3.97 0.15 1.01 0.76 0.99 0.61 2.48 3.54 3.76 0.78 3.12 2.58 2.63 }

Matrix C:
{ 70.80 70.44 60.55 55.63 53.52 71.79 58.92 74.93 70.70 69.21 45.94 61.33 58.39 76.84 39.43 51.63 }
{ 49.17 55.72 39.75 45.64 41.92 50.29 33.63 54.18 54.02 49.74 42.96 47.97 38.99 63.23 32.08 42.80 }
{ 75.95 76.97 64.24 58.84 65.36 87.00 65.14 83.46 76.66 76.39 53.56 62.77 57.28 88.26 47.16 62.48 }
{ 81.42 72.16 62.48 61.42 69.52 80.93 66.88 74.54 61.90 67.46 55.16 64.62 55.39 89.55 51.41 72.63 }
{ 77.62 83.65 58.76 56.78 68.43 71.75 58.12 76.98 66.41 77.38 51.61 68.67 54.89 88.44 48.11 60.74 }
{ 71.03 68.16 63.49 70.94 66.00 76.27 55.70 78.00 72.34 68.76 58.16 63.07 59.81 82.01 35.69 65.14 }
{ 65.69 67.09 63.95 69.06 62.88 71.89 68.78 67.00 56.89 69.80 46.87 77.88 55.18 88.18 47.56 66.43 }
{ 71.27 68.53 68.56 63.22 66.02 72.17 58.20 72.37 73.04 83.53 55.71 76.59 52.55 79.42 37.49 64.46 }
{ 65.70 70.98 62.60 48.89 67.59 73.68 51.46 77.55 62.28 64.62 53.95 66.86 54.37 81.99 35.07 60.77 }
{ 81.04 68.29 71.31 74.53 71.75 80.50 55.19 80.01 72.40 75.56 58.60 71.48 54.67 86.42 39.28 60.14 }
{ 70.25 64.43 50.70 67.98 61.70 68.88 41.78 63.04 61.80 61.66 58.96 55.29 53.30 74.84 39.85 63.85 }
{ 56.59 49.83 47.54 48.05 48.80 53.69 56.22 58.36 59.48 50.12 43.27 52.23 49.73 61.82 24.73 50.78 }
{ 63.07 65.14 60.88 64.05 63.59 68.71 50.39 67.48 56.20 74.25 57.66 64.72 49.31 77.56 38.70 69.60 }
{ 74.40 75.85 63.54 58.34 60.90 70.58 62.65 70.61 52.94 80.12 45.19 66.55 56.93 70.47 41.53 70.33 }
{ 77.63 83.83 66.00 64.84 72.50 80.28 59.14 85.22 76.03 79.55 57.21 69.55 60.86 90.57 43.11 64.04 }
{ 67.19 74.27 59.60 60.77 65.49 64.51 62.49 72.07 63.26 70.87 56.01 72.25 59.15 87.55 44.04 64.86 }
```

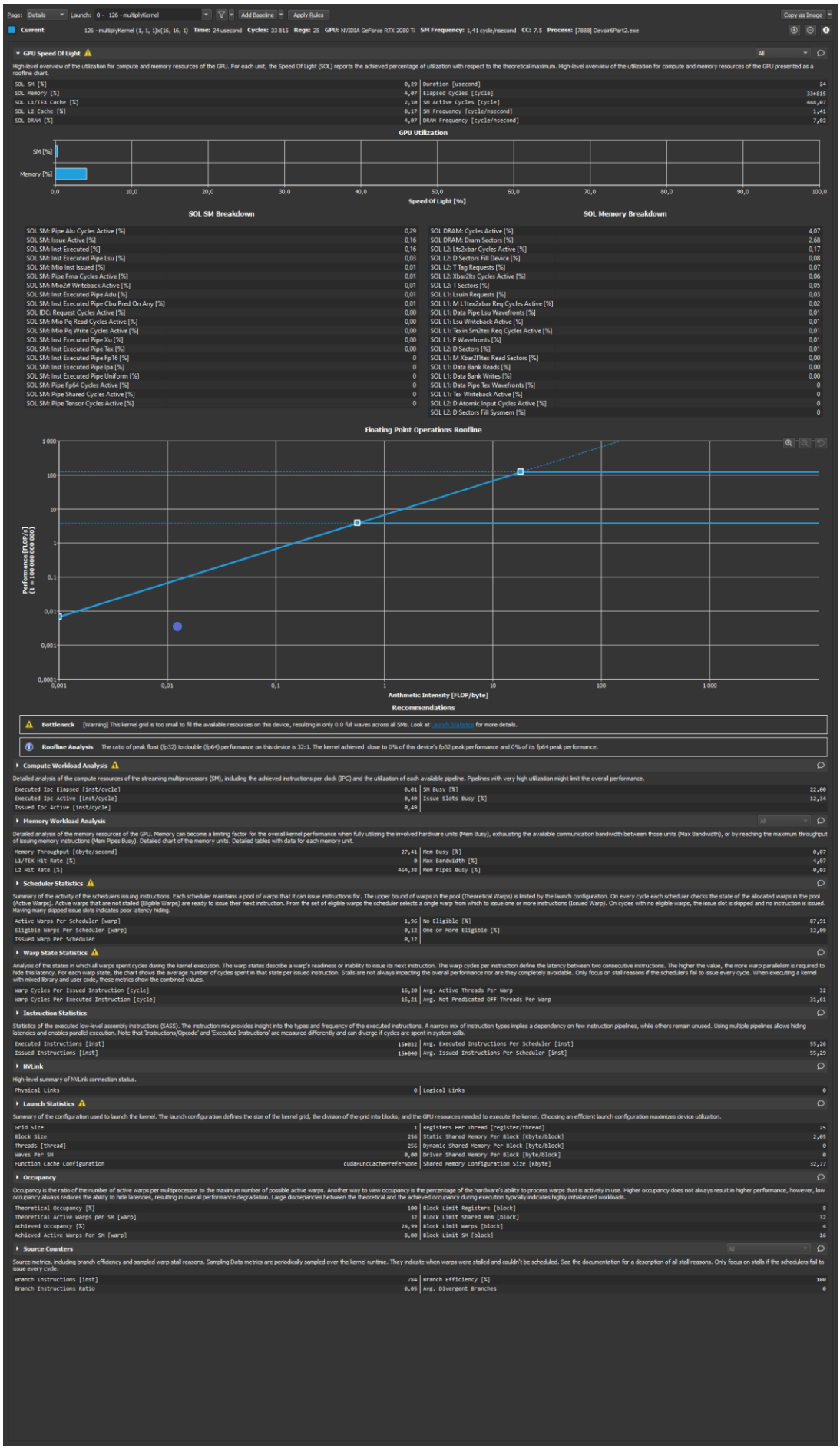
4. Profilez chaque implementation.

Tuile 2*2 :

Nvprof :

```
==23080== Profiling application: a
==23080== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   55.43%  4.8960us         1  4.8960us  4.8960us  4.8960us  multiplyKernel(float*, float const *, float const *, int)
                   33.70%  2.9760us         2  1.4880us  1.1520us  1.8240us  [CUDA memcpy HtoD]
                   10.87%     960ns         1     960ns     960ns     960ns  [CUDA memcpy DtoH]
      API calls:   79.58%  200.94ms         3  66.980ms  5.7000us  200.93ms  cudaMalloc
                   20.18%  50.947ms         1  50.947ms  50.947ms  50.947ms  cudaDeviceReset
                    0.13%  330.50us         3  110.17us  51.500us  218.00us  cudaMemcpy
                    0.05%  119.20us         3  39.733us  1.9000us  112.50us  cudaFree
                    0.03%  68.100us         1  68.100us  68.100us  68.100us  cudaLaunchKernel
                    0.02%  43.000us         1  43.000us  43.000us  43.000us  cudaDeviceSynchronize
                    0.01%  15.600us       101     154ns     100ns  1.2000us  cuDeviceGetAttribute
                    0.01%  14.200us         3  4.7330us     200ns  13.500us  cuDeviceGetCount
                    0.00%  6.1000us         1  6.1000us  6.1000us  6.1000us  cudaSetDevice
                    0.00%  1.7000us         2     850ns     200ns  1.5000us  cuDeviceGet
                    0.00%  1.1000us         1  1.1000us  1.1000us  1.1000us  cuDeviceGetName
                    0.00%     400ns         1     400ns     400ns     400ns  cuDeviceGetLuid
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceTotalMem
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceGetUuid
                    0.00%     200ns         1     200ns     200ns     200ns  cudaGetLastError

C:\Users\lacommap\Desktop\CudaC\Devoir6Part2>
```

Nsight :

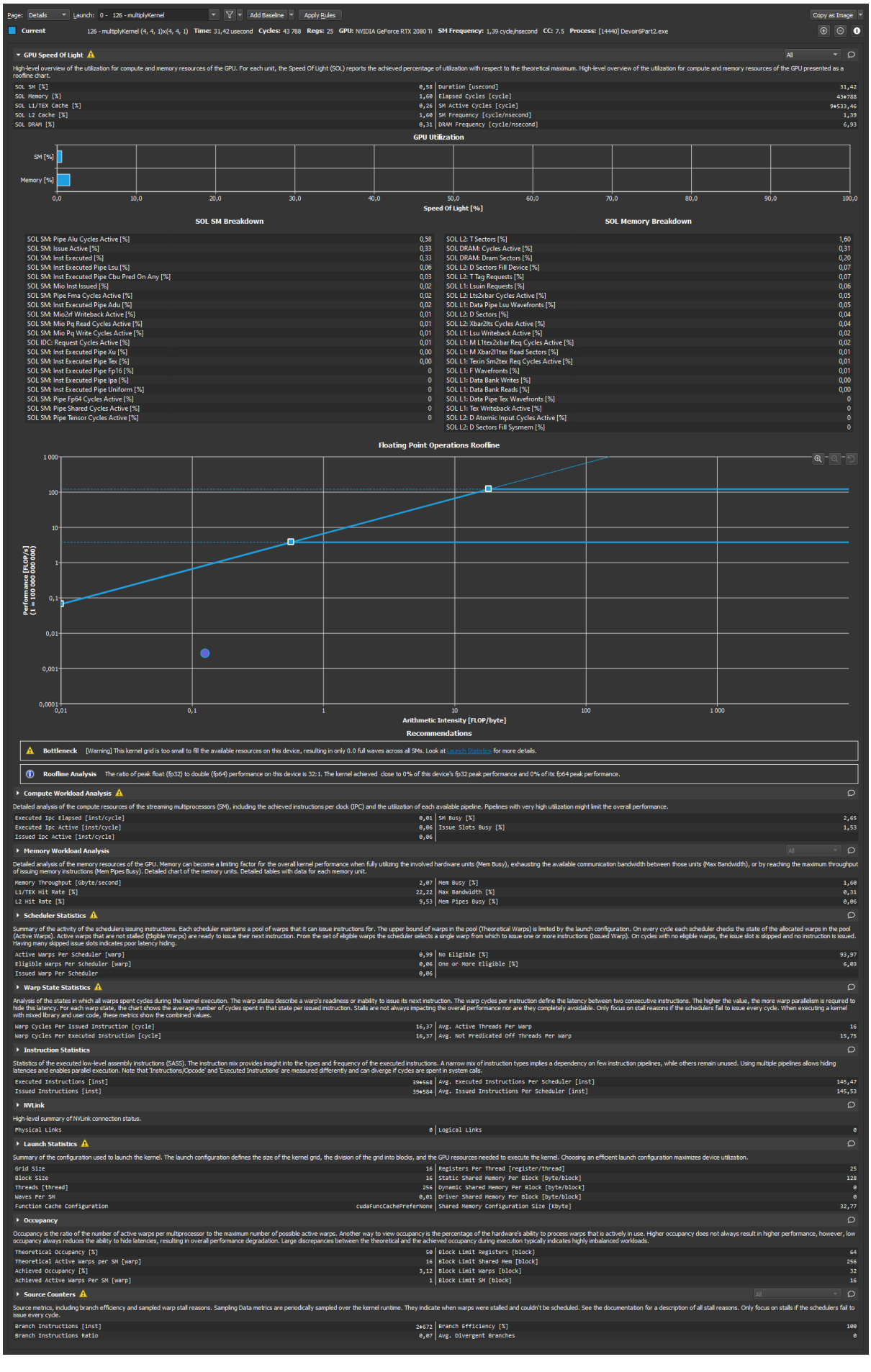

Page: Details
Launch: 0 - 126 - multiplyKernel
Add Baseline
Apply Rules
Copy as Image
Current
126 - multiplyKernel (1, 1, 1)x(16, 16, 1) Time: 24 usecond Cycles: 33 815 Regs: 25 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,41 cycle/nsecond CC: 7.5 Process: [7888] DevoirGPart2.exe
GPU Speed Of Light
High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.
SOL SM [%] 0,29
SOL Memory [%] 4,07
SOL L1/TEX Cache [%] 2,18
SOL L2 Cache [%] 0,17
SOL DRAM [%] 4,07
Duration [usecond] 24
Elapsed Cycles [cycle] 33•815
SM Active Cycles [cycle] 448,07
SM Frequency [cycle/nsecond] 1,41
DRAM Frequency [cycle/nsecond] 7,02
GPU Utilization
SM [%]
Memory [%]
0,0 10,0 20,0 30,0 40,0 50,0 60,0 70,0 80,0 90,0 100,0
Speed Of Light [%]
SOL SM Breakdown
SOL SM: Pipe Alu Cycles Active [%] 0,29
SOL SM: Issue Active [%] 0,16
SOL SM: Inst Executed [%] 0,16
SOL SM: Inst Executed Pipe Lsu [%] 0,03
SOL SM: Mio Inst Issued [%] 0,01
SOL SM: Pipe Fma Cycles Active [%] 0,01
SOL SM: Mio2rf Writeback Active [%] 0,01
SOL SM: Inst Executed Pipe Adiu [%] 0,01
SOL SM: Inst Executed Pipe Cbu Pred On Any [%] 0,01
SOL IDC: Request Cycles Active [%] 0,00
SOL SM: Mio Pq Read Cycles Active [%] 0,00
SOL SM: Mio Pq Write Cycles Active [%] 0,00
SOL SM: Inst Executed Pipe Xu [%] 0,00
SOL SM: Inst Executed Pipe Tex [%] 0,00
SOL SM: Inst Executed Pipe Fp16 [%] 0
SOL SM: Inst Executed Pipe Ipa [%] 0
SOL SM: Inst Executed Pipe Uniform [%] 0
SOL SM: Pipe Fp64 Cycles Active [%] 0
SOL SM: Pipe Shared Cycles Active [%] 0
SOL SM: Pipe Tensor Cycles Active [%] 0
SOL Memory Breakdown
SOL DRAM: Cycles Active [%] 4,07
SOL DRAM: Dram Sectors [%] 2,68
SOL L2: Lts2xbar Cycles Active [%] 0,17
SOL L2: D Sectors Fill Device [%] 0,08
SOL L2: T Tag Requests [%] 0,07
SOL L2: Xbar2lts Cycles Active [%] 0,06
SOL L2: T Sectors [%] 0,05
SOL L1: Lsuin Requests [%] 0,03
SOL L1: M L1tex2xbar Req Cycles Active [%] 0,02
SOL L1: Data Pipe Lsu Wavefronts [%] 0,01
SOL L1: Lsu Writeback Active [%] 0,01
SOL L1: Texin Sm2tex Req Cycles Active [%] 0,01
SOL L1: F Wavefronts [%] 0,01
SOL L2: D Sectors [%] 0,01
SOL L1: M Xbar2l1tex Read Sectors [%] 0,00
SOL L1: Data Bank Reads [%] 0,00
SOL L1: Data Bank Writes [%] 0,00
SOL L1: Data Pipe Tex Wavefronts [%] 0
SOL L1: Tex Writeback Active [%] 0
SOL L2: D Atomic Input Cycles Active [%] 0
SOL L2: D Sectors Fill Sysmem [%] 0
Floating Point Operations Roofline
Performance [FLOP/s] (1 : 1 000 000 000 000)
1 000 100 10 1 0,1 0,01 0,001 0,0001
0,001 0,01 0,1 1 10 100 1 000
Arithmetic Intensity [FLOP/byte]
Recommendations
Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.0 full waves across all SMs. Look at Launch Statistics for more details.
Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.
Compute Workload Analysis
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
Executed Ipc Elapsed [inst/cycle] 0,01
Executed Ipc Active [inst/cycle] 0,49
Issued Ipc Active [inst/cycle] 0,43
SM Busy [%] 22,00
Issue Slots Busy [%] 12,34
Memory Workload Analysis
Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.
Memory Throughput [Gbyte/second] 27,41
L1/TEX Hit Rate [%] 0
L2 Hit Rate [%] 464,38
Mem Busy [%] 0,07
Max Bandwidth [%] 4,07
Mem Pipes Busy [%] 0,03
Scheduler Statistics
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.
Active Warps Per Scheduler [warp] 1,96
Eligible Warps Per Scheduler [warp] 0,12
Issued Warp Per Scheduler 0,12
No Eligible [%] 87,91
One or More Eligible [%] 12,09
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.
Warp Cycles Per Issued Instruction [cycle] 16,20
Warp Cycles Per Executed Instruction [cycle] 16,21
Avg. Active Threads Per Warp 32
Avg. Not Predicated Off Threads Per Warp 31,61
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if codes are spent in system calls.
Executed Instructions [inst] 15•032
Issued Instructions [inst] 15•040
Avg. Executed Instructions Per Scheduler [inst] 55,26
Avg. Issued Instructions Per Scheduler [inst] 55,29
NVLink
High-level summary of NVLink connection status.
Physical Links 0
Logical Links 0
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 1
Block Size 256
Threads [thread] 256
Waves Per SM 0,00
Function Cache Configuration cudaFuncCachePreferNone
Registers Per Thread [register/thread] 25
Static Shared Memory Per Block [Kbyte/block] 2,05
Dynamic Shared Memory Per Block [byte/block] 0
Driver Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [Kbyte] 32,77
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 100
Theoretical Active Warps per SM [warp] 32
Achieved Occupancy [%] 24,99
Achieved Active Warps Per SM [warp] 8,00
Block Limit Registers [block] 8
Block Limit Shared Mem [block] 32
Block Limit Warps [block] 4
Block Limit SM [block] 16
Source Counters
Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.
Branch Instructions [inst] 784
Branch Instructions Ratio 0,05
Branch Efficiency [%] 100
Avg. Divergent Branches 0

Tuile 4*4 :

Nvprof :

```
==6336== Profiling application: a
==6336== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   52.20%  4.1600us         1  4.1600us  4.1600us  4.1600us  multiplyKernel(float*, float const *, float const *, int)
                   35.35%  2.8170us         2  1.4080us  1.1210us  1.6960us  [CUDA memcpy HtoD]
                   12.45%     992ns         1     992ns     992ns     992ns  [CUDA memcpy DtoH]
      API calls:   79.29%  198.80ms         3  66.265ms  2.0000us  198.79ms  cudaMalloc
                   20.48%  51.348ms         1  51.348ms  51.348ms  51.348ms  cudaDeviceReset
                    0.14%  345.90us         3  115.30us  34.000us  259.20us  cudaMemcpy
                    0.05%  117.90us         3  39.300us  1.9000us  111.70us  cudaFree
                    0.02%  53.900us         1  53.900us  53.900us  53.900us  cudaLaunchKernel
                    0.02%  40.700us         1  40.700us  40.700us  40.700us  cudaDeviceSynchronize
                    0.01%  14.800us       101     146ns     100ns  1.0000us  cuDeviceGetAttribute
                    0.00%  6.3000us         1  6.3000us  6.3000us  6.3000us  cudaSetDevice
                    0.00%  6.1000us         3  2.0330us     300ns  5.4000us  cuDeviceGetCount
                    0.00%  1.8000us         2     900ns     100ns  1.7000us  cuDeviceGet
                    0.00%     800ns         1     800ns     800ns     800ns  cuDeviceGetName
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceTotalMem
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceGetLuid
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceGetUuid
                    0.00%     200ns         1     200ns     200ns     200ns  cudaGetLastError

C:\Users\lacommap\Desktop\CudaC\Devoir6Part2>
```

Nsight :

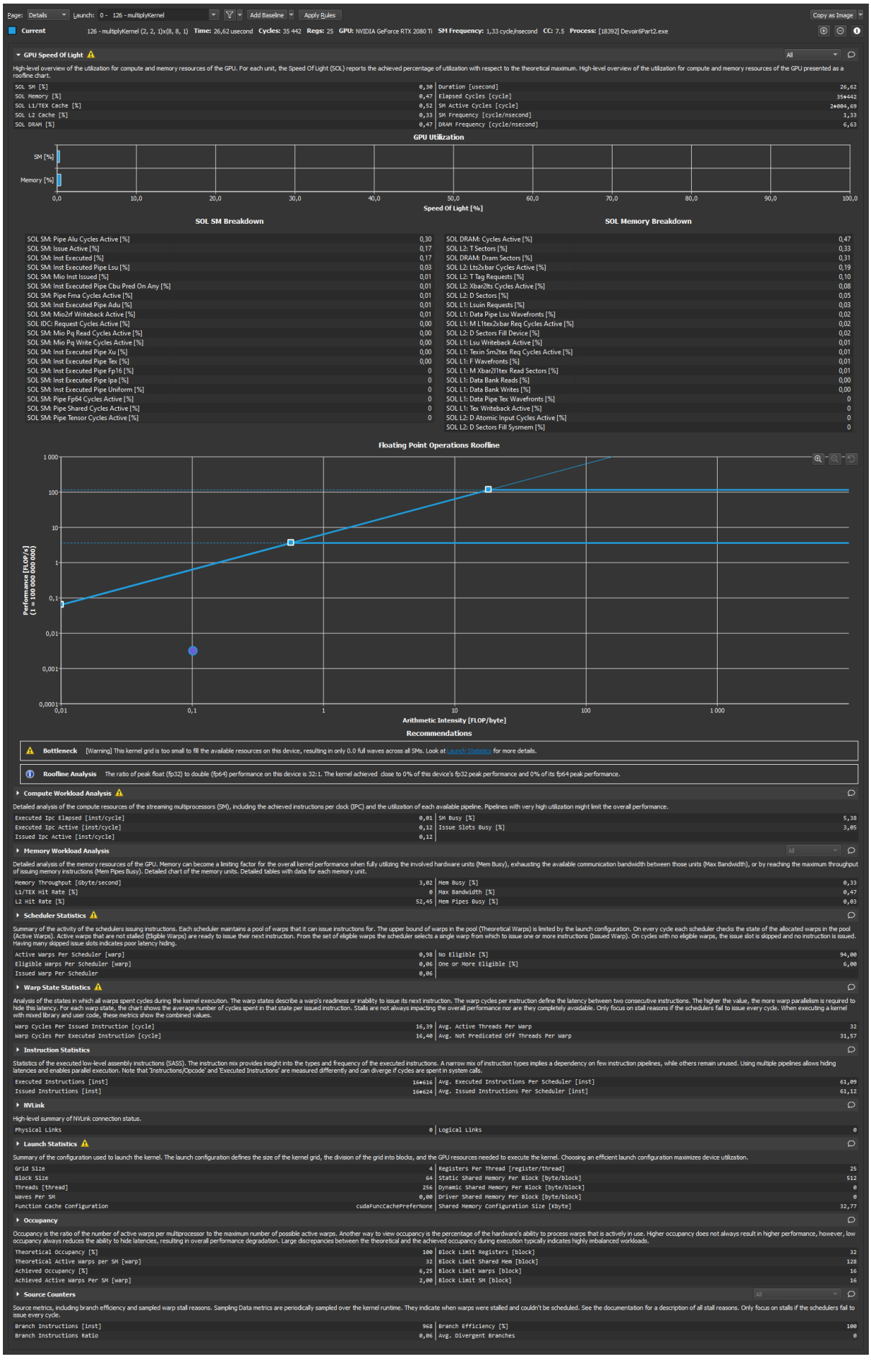

Page: Details
Launch: 0 - 126 - multiplyKernel
Add Baseline
Apply Rules
Copy as Image
Current
126 - multiplyKernel (4, 4, 1)x(4, 4, 1) Time: 31,42 usecond Cycles: 43 788 Regs: 25 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,39 cycle/nsecond CC: 7.5 Process: [14440] Devoir6Part2.exe
GPU Speed Of Light
High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.
SOL SM [%] 0,58
SOL Memory [%] 1,60
SOL L1/TEX Cache [%] 0,26
SOL L2 Cache [%] 1,60
SOL DRAM [%] 0,31
Duration [usecond] 31,42
Elapsed Cycles [cycle] 43 788
SM Active Cycles [cycle] 9 533,46
SM Frequency [cycle/nsecond] 1,39
DRAM Frequency [cycle/nsecond] 6,93
GPU Utilization
SM [%]
Memory [%]
0,0
10,0
20,0
30,0
40,0
50,0
60,0
70,0
80,0
90,0
100,0
Speed Of Light [%]
SOL SM Breakdown
SOL SM: Pipe Alu Cycles Active [%] 0,58
SOL SM: Issue Active [%] 0,33
SOL SM: Inst Executed [%] 0,33
SOL SM: Inst Executed Pipe Lsu [%] 0,06
SOL SM: Inst Executed Pipe Cbu Pred On Any [%] 0,03
SOL SM: Mio Inst Issued [%] 0,02
SOL SM: Pipe Fma Cycles Active [%] 0,02
SOL SM: Inst Executed Pipe Adu [%] 0,02
SOL SM: Mio2rf Writeback Active [%] 0,01
SOL SM: Mio Pq Read Cycles Active [%] 0,01
SOL SM: Mio Pq Write Cycles Active [%] 0,01
SOL IDC: Request Cycles Active [%] 0,01
SOL SM: Inst Executed Pipe Xu [%] 0,00
SOL SM: Inst Executed Pipe Tex [%] 0,00
SOL SM: Inst Executed Pipe Fp16 [%] 0
SOL SM: Inst Executed Pipe Ipa [%] 0
SOL SM: Inst Executed Pipe Uniform [%] 0
SOL SM: Pipe Fp64 Cycles Active [%] 0
SOL SM: Pipe Shared Cycles Active [%] 0
SOL SM: Pipe Tensor Cycles Active [%] 0
SOL Memory Breakdown
SOL L2: T Sectors [%] 1,60
SOL DRAM: Cycles Active [%] 0,31
SOL DRAM: Dram Sectors [%] 0,20
SOL L2: D Sectors Fill Device [%] 0,07
SOL L2: T Tag Requests [%] 0,07
SOL L1: Lsuin Requests [%] 0,06
SOL L2: Lts2xbar Cycles Active [%] 0,05
SOL L1: Data Pipe Lsu Wavefronts [%] 0,05
SOL L2: D Sectors [%] 0,04
SOL L2: Xbar2lts Cycles Active [%] 0,04
SOL L1: Lsu Writeback Active [%] 0,02
SOL L1: M L1tex2xbar Req Cycles Active [%] 0,02
SOL L1: M Xbar2l1tex Read Sectors [%] 0,01
SOL L1: Texin Sm2tex Req Cycles Active [%] 0,01
SOL L1: F Wavefronts [%] 0,01
SOL L1: Data Bank Writes [%] 0,00
SOL L1: Data Bank Reads [%] 0,00
SOL L1: Data Pipe Tex Wavefronts [%] 0
SOL L1: Tex Writeback Active [%] 0
SOL L2: D Atomic Input Cycles Active [%] 0
SOL L2: D Sectors Fill Sysmem [%] 0
Floating Point Operations Roofline
Performance [FLOP/s] (1 = 100 000 000 000)
1 000
100
10
1
0,1
0,01
0,001
0,0001
0,01
0,1
1
10
100
1 000
Arithmetic Intensity [FLOP/byte]
Recommendations
Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.0 full waves across all SMs. Look at Launch Statistics for more details.
Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.
Compute Workload Analysis
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
Executed Ipc Elapsed [inst/cycle] 0,01
Executed Ipc Active [inst/cycle] 0,06
Issued Ipc Active [inst/cycle] 0,06
SM Busy [%] 2,65
Issue Slots Busy [%] 1,53
Memory Workload Analysis
Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.
Memory Throughput [Gbyte/second] 2,07
L1/TEX Hit Rate [%] 22,22
L2 Hit Rate [%] 9,53
Mem Busy [%] 1,60
Max Bandwidth [%] 0,31
Mem Pipes Busy [%] 0,06
Scheduler Statistics
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.
Active Warps Per Scheduler [warp] 0,99
Eligible Warps Per Scheduler [warp] 0,06
Issued Warp Per Scheduler 0,06
No Eligible [%] 93,97
One or More Eligible [%] 6,03
Warp State Statistics
Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.
Warp Cycles Per Issued Instruction [cycle] 16,37
Warp Cycles Per Executed Instruction [cycle] 16,37
Avg. Active Threads Per Warp 16
Avg. Not Predicated Off Threads Per Warp 15,75
Instruction Statistics
Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.
Executed Instructions [inst] 39 568
Issued Instructions [inst] 39 584
Avg. Executed Instructions Per Scheduler [inst] 145,47
Avg. Issued Instructions Per Scheduler [inst] 145,53
NVLink
High-level summary of NVLink connection status.
Physical Links 0
Logical Links 0
Launch Statistics
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.
Grid Size 16
Block Size 16
Threads [thread] 256
Waves Per SM 0,01
Function Cache Configuration cudaFuncCachePreferNone
Registers Per Thread [register/thread] 25
Static Shared Memory Per Block [byte/block] 128
Dynamic Shared Memory Per Block [byte/block] 0
Driver Shared Memory Per Block [byte/block] 0
Shared Memory Configuration Size [kbyte] 32,77
Occupancy
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.
Theoretical Occupancy [%] 50
Theoretical Active Warps per SM [warp] 16
Achieved Occupancy [%] 3,12
Achieved Active Warps Per SM [warp] 1
Block Limit Registers [block] 64
Block Limit Shared Mem [block] 256
Block Limit Warps [block] 32
Block Limit SM [block] 16
Source Counters
Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.
Branch Instructions [inst] 2 672
Branch Instructions Ratio 0,07
Branch Efficiency [%] 100
Avg. Divergent Branches 0

Tuile 8*8 :

Nvprof :

```
==21300== Profiling application: a
==21300== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   47.06%  3.5840us         1  3.5840us  3.5840us  3.5840us  multiplyKernel(float*, float const *, float const *, int)
                   40.76%  3.1040us         2  1.5520us  1.1200us  1.9840us  [CUDA memcpy HtoD]
                   12.18%     928ns         1     928ns     928ns     928ns  [CUDA memcpy DtoH]
      API calls:   79.02%  198.83ms         3  66.275ms  1.8000us  198.82ms  cudaMalloc
                   20.77%  52.262ms         1  52.262ms  52.262ms  52.262ms  cudaDeviceReset
                    0.11%  285.10us         3  95.033us  33.900us  201.60us  cudaMemcpy
                    0.05%  114.50us         3  38.166us  1.8000us  108.30us  cudaFree
                    0.02%  52.800us         1  52.800us  52.800us  52.800us  cudaLaunchKernel
                    0.02%  40.000us         1  40.000us  40.000us  40.000us  cudaDeviceSynchronize
                    0.01%  14.200us       101     140ns     100ns     800ns  cuDeviceGetAttribute
                    0.00%  5.9000us         3  1.9660us     200ns  5.3000us  cuDeviceGetCount
                    0.00%  5.5000us         1  5.5000us  5.5000us  5.5000us  cudaSetDevice
                    0.00%  1.4000us         2     700ns     100ns  1.3000us  cuDeviceGet
                    0.00%     600ns         1     600ns     600ns     600ns  cuDeviceGetName
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceTotalMem
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceGetLuid
                    0.00%     200ns         1     200ns     200ns     200ns  cudaGetLastError
                    0.00%     100ns         1     100ns     100ns     100ns  cuDeviceGetUuid

C:\Users\lacommap\Desktop\CudaC\Devoir6Part2>
```

Nsight :



Page: Details | Launch: 0 - 126 - multiplyKernel | Add Baseline | Apply Rules | Copy as Image

Current 126 - multiplyKernel (2, 2, 1)x(8, 8, 1) Time: 26,62 usecond Cycles: 35 442 Regs: 25 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,33 cycle/nsecond CC: 7.5 Process: [18392] Devoir6Part2.exe

GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	0,30	Duration [usecond]	26,62
SOL Memory [%]	0,47	Elapsed Cycles [cycle]	35442
SOL L1/TEX Cache [%]	0,52	SM Active Cycles [cycle]	24084,69
SOL L2 Cache [%]	0,33	SM Frequency [cycle/nsecond]	1,33
SOL DRAM [%]	0,47	DRAM Frequency [cycle/nsecond]	6,63

GPU Utilization

SOL SM Breakdown

SOL SM: Pipe Alu Cycles Active [%]	0,30
SOL SM: Issue Active [%]	0,17
SOL SM: Inst Executed [%]	0,17
SOL SM: Inst Executed Pipe Lsu [%]	0,03
SOL SM: Mio Inst Issued [%]	0,01
SOL SM: Inst Executed Pipe Cbu Pred On Any [%]	0,01
SOL SM: Pipe Fma Cycles Active [%]	0,01
SOL SM: Inst Executed Pipe Adu [%]	0,01
SOL SM: Mio2rf Writeback Active [%]	0,01
SOL IDC: Request Cycles Active [%]	0,00
SOL SM: Mio Pq Read Cycles Active [%]	0,00
SOL SM: Mio Pq Write Cycles Active [%]	0,00
SOL SM: Inst Executed Pipe Xu [%]	0,00
SOL SM: Inst Executed Pipe Tex [%]	0,00
SOL SM: Inst Executed Pipe Fp16 [%]	0
SOL SM: Inst Executed Pipe Ipa [%]	0
SOL SM: Inst Executed Pipe Uniform [%]	0
SOL SM: Pipe Fp64 Cycles Active [%]	0
SOL SM: Pipe Shared Cycles Active [%]	0
SOL SM: Pipe Tensor Cycles Active [%]	0

SOL Memory Breakdown

SOL DRAM: Cycles Active [%]	0,47
SOL L2: T Sectors [%]	0,33
SOL DRAM: Dram Sectors [%]	0,31
SOL L2: Lts2xbar Cycles Active [%]	0,19
SOL L2: T Tag Requests [%]	0,10
SOL L2: Xbar2lts Cycles Active [%]	0,08
SOL L2: D Sectors [%]	0,05
SOL L1: Lsuin Requests [%]	0,03
SOL L1: Data Pipe Lsu Wavefronts [%]	0,02
SOL L1: M L1tex2xbar Req Cycles Active [%]	0,02
SOL L2: D Sectors Fill Device [%]	0,02
SOL L1: Lsu Writeback Active [%]	0,01
SOL L1: Texin Sm2tex Req Cycles Active [%]	0,01
SOL L1: F Wavefronts [%]	0,01
SOL L1: M Xbar2l1tex Read Sectors [%]	0,01
SOL L1: Data Bank Reads [%]	0,00
SOL L1: Data Bank Writes [%]	0,00
SOL L1: Data Pipe Tex Wavefronts [%]	0
SOL L1: Tex Writeback Active [%]	0
SOL L2: D Atomic Input Cycles Active [%]	0
SOL L2: D Sectors Fill Sysmem [%]	0

Floating Point Operations Roofline

Recommendations

Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.0 full waves across all SMs. Look at Launch Statistics for more details.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	0,01	SM Busy [%]	5,38
Executed Ipc Active [inst/cycle]	0,12	Issue Slots Busy [%]	3,05
Issued Ipc Active [inst/cycle]	0,12		

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	3,02	Mem Busy [%]	0,33
L1/TEX Hit Rate [%]	0	Max Bandwidth [%]	0,47
L2 Hit Rate [%]	52,45	Mem Pipes Busy [%]	0,03

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	0,98	No Eligible [%]	94,00
Eligible Warps Per Scheduler [warp]	0,06	One or More Eligible [%]	6,00
Issued Warp Per Scheduler	0,06		

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	16,39	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	16,40	Avg. Not Predicated Off Threads Per Warp	31,57

Instruction Statistics

Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

Executed Instructions [inst]	16616	Avg. Executed Instructions Per Scheduler [inst]	61,09
Issued Instructions [inst]	16624	Avg. Issued Instructions Per Scheduler [inst]	61,12

NVLink

High-level summary of NVLink connection status.

Physical Links	0	Logical Links	0

Launch Statistics

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	4	Registers Per Thread [register/thread]	25
Block Size	64	Static Shared Memory Per Block [byte/block]	512
Threads [thread]	256	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	0,00	Driver Shared Memory Per Block [byte/block]	0
Function Cache Configuration	cudaFuncCachePreferNone	Shared Memory Configuration Size [kbyte]	32,77

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	100	Block Limit Registers [block]	32
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	128
Achieved Occupancy [%]	6,25	Block Limit Warps [block]	16
Achieved Active Warps Per SM [warp]	2,00	Block Limit SM [block]	16

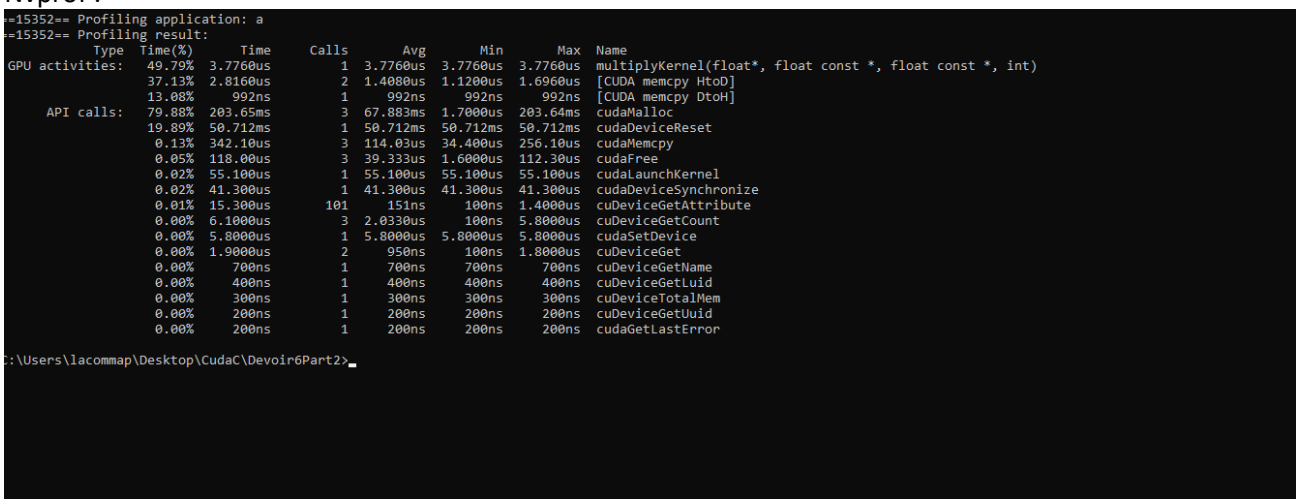
Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.

Branch Instructions [inst]	968	Branch Efficiency [%]	100
Branch Instructions Ratio	0,06	Avg. Divergent Branches	0

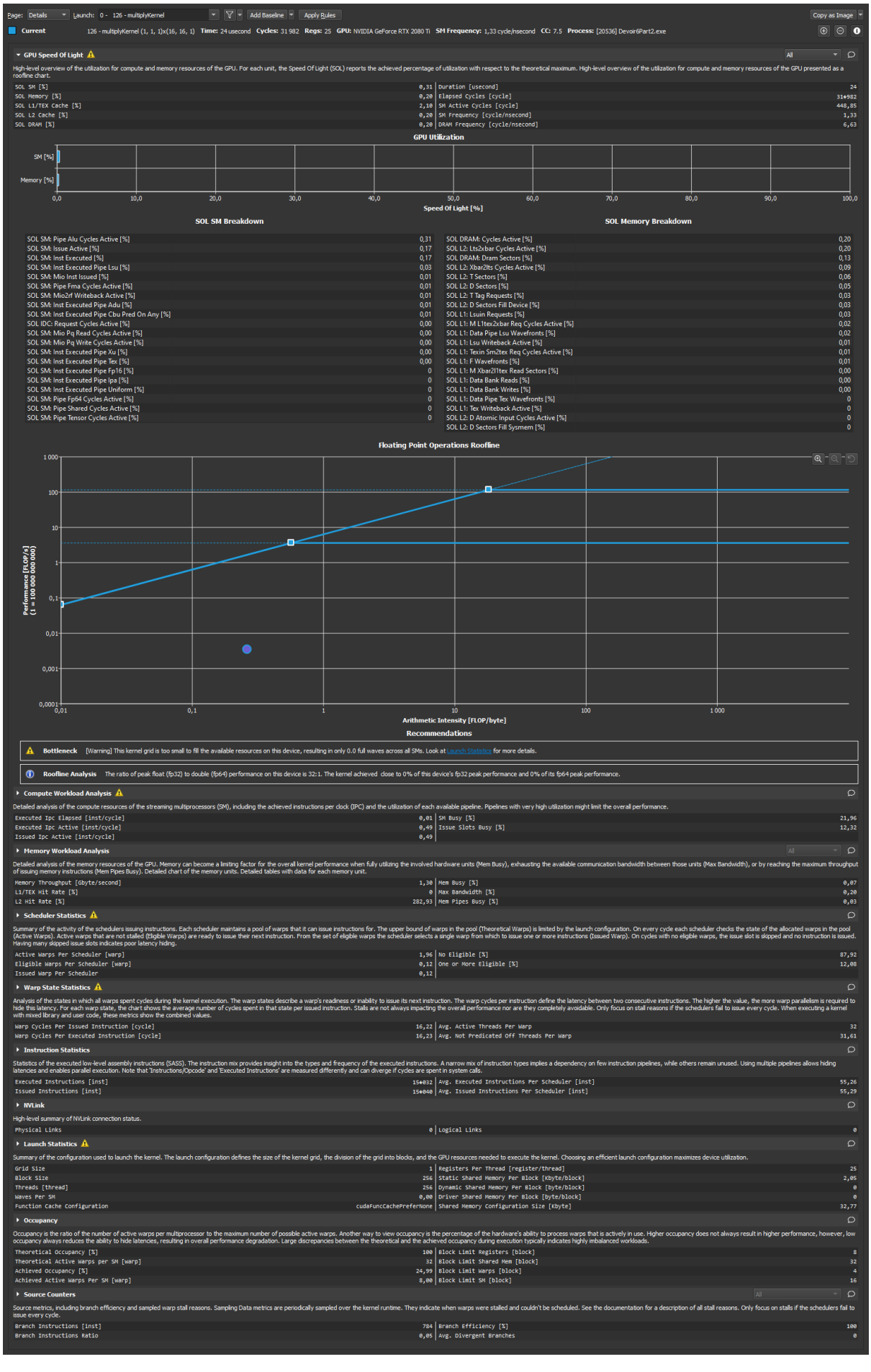
Tuile 16*16 :

Nvprof :



```
==15352== Profiling application: a
==15352== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   49.79%  3.7760us         1  3.7760us  3.7760us  3.7760us  multiplyKernel(float*, float const *, float const *, int)
                   37.13%  2.8160us         2  1.4080us  1.1200us  1.6960us  [CUDA memcpy HtoD]
                   13.08%     992ns         1     992ns     992ns     992ns  [CUDA memcpy DtoH]
      API calls:   79.88%  203.65ms         3  67.883ms  1.7000us  203.64ms  cudaMalloc
                   19.89%  50.712ms         1  50.712ms  50.712ms  50.712ms  cudaDeviceReset
                    0.13%  342.10us         3  114.03us  34.400us  256.10us  cudaMemcpy
                    0.05%  118.00us         3  39.333us  1.6000us  112.30us  cudaFree
                    0.02%  55.100us         1  55.100us  55.100us  55.100us  cudaLaunchKernel
                    0.02%  41.300us         1  41.300us  41.300us  41.300us  cudaDeviceSynchronize
                    0.01%  15.300us       101     151ns     100ns  1.4000us  cuDeviceGetAttribute
                    0.00%  6.1000us         3  2.0330us     100ns  5.8000us  cuDeviceGetCount
                    0.00%  5.8000us         1  5.8000us  5.8000us  5.8000us  cudaSetDevice
                    0.00%  1.9000us         2     950ns     100ns  1.8000us  cuDeviceGet
                    0.00%     700ns         1     700ns     700ns     700ns  cuDeviceGetName
                    0.00%     400ns         1     400ns     400ns     400ns  cuDeviceGetLuid
                    0.00%     300ns         1     300ns     300ns     300ns  cuDeviceTotalMem
                    0.00%     200ns         1     200ns     200ns     200ns  cuDeviceGetUuid
                    0.00%     200ns         1     200ns     200ns     200ns  cudaGetLastError

C:\Users\lacommap\Desktop\CudaC\Devoir6Part2>
```

Nsight :



Page: Details | Launch: 0 - 126 - multiplyKernel | Add Baseline | Apply Rules | Copy as Image

Current 126 - multiplyKernel (1, 1, 1)x(16, 16, 1) Time: 24 usecond Cycles: 31 982 Regs: 25 GPU: NVIDIA GeForce RTX 2080 Ti SM Frequency: 1,33 cycle/nsecond CC: 7.5 Process: [20536] Devoir6Part2.exe

GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	0,31	Duration [usecond]	24
SOL Memory [%]	0,20	Elapsed Cycles [cycle]	31+982
SOL L1/TEX Cache [%]	2,10	SM Active Cycles [cycle]	448,85
SOL L2 Cache [%]	0,20	SM Frequency [cycle/nsecond]	1,33
SOL DRAM [%]	0,20	DRAM Frequency [cycle/nsecond]	6,63

GPU Utilization

SOL SM Breakdown

SOL SM: Pipe Alu Cycles Active [%]	0,31
SOL SM: Issue Active [%]	0,17
SOL SM: Inst Executed [%]	0,17
SOL SM: Inst Executed Pipe Lsu [%]	0,03
SOL SM: Mio Inst Issued [%]	0,01
SOL SM: Pipe Fma Cycles Active [%]	0,01
SOL SM: Mio2rf Writeback Active [%]	0,01
SOL SM: Inst Executed Pipe Adu [%]	0,01
SOL SM: Inst Executed Pipe Cbu Pred On Any [%]	0,01
SOL IDC: Request Cycles Active [%]	0,00
SOL SM: Mio Pq Read Cycles Active [%]	0,00
SOL SM: Mio Pq Write Cycles Active [%]	0,00
SOL SM: Inst Executed Pipe Xu [%]	0,00
SOL SM: Inst Executed Pipe Tex [%]	0,00
SOL SM: Inst Executed Pipe Fp16 [%]	0
SOL SM: Inst Executed Pipe Ipa [%]	0
SOL SM: Inst Executed Pipe Uniform [%]	0
SOL SM: Pipe Fp64 Cycles Active [%]	0
SOL SM: Pipe Shared Cycles Active [%]	0
SOL SM: Pipe Tensor Cycles Active [%]	0

SOL Memory Breakdown

SOL DRAM: Cycles Active [%]	0,20
SOL L2: Lts2xbar Cycles Active [%]	0,20
SOL DRAM: Dram Sectors [%]	0,13
SOL L2: Xbar2lts Cycles Active [%]	0,09
SOL L2: T Sectors [%]	0,06
SOL L2: D Sectors [%]	0,05
SOL L2: T Tag Requests [%]	0,03
SOL L2: D Sectors Fill Device [%]	0,03
SOL L1: Lsuin Requests [%]	0,03
SOL L1: M L1tex2xbar Req Cycles Active [%]	0,02
SOL L1: Data Pipe Lsu Wavefronts [%]	0,02
SOL L1: Lsu Writeback Active [%]	0,01
SOL L1: Texin Sm2tex Req Cycles Active [%]	0,01
SOL L1: F Wavefronts [%]	0,01
SOL L1: M Xbar2l1tex Read Sectors [%]	0,00
SOL L1: Data Bank Reads [%]	0,00
SOL L1: Data Bank Writes [%]	0,00
SOL L1: Data Pipe Tex Wavefronts [%]	0
SOL L1: Tex Writeback Active [%]	0
SOL L2: D Atomic Input Cycles Active [%]	0
SOL L2: D Sectors Fill Sysmem [%]	0

Floating Point Operations Roofline

Recommendations

Bottleneck [Warning] This kernel grid is too small to fill the available resources on this device, resulting in only 0.0 full waves across all SMs. Look at Launch Statistics for more details.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 32:1. The kernel achieved close to 0% of this device's fp32 peak performance and 0% of its fp64 peak performance.

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	0,01	SM Busy [%]	21,96
Executed Ipc Active [inst/cycle]	0,49	Issue Slots Busy [%]	12,32
Issued Ipc Active [inst/cycle]	0,49		

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	1,30	Mem Busy [%]	0,07
L1/TEX Hit Rate [%]	0	Max Bandwidth [%]	0,20
L2 Hit Rate [%]	282,93	Mem Pipes Busy [%]	0,03

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	1,96	No Eligible [%]	87,92
Eligible Warps Per Scheduler [warp]	0,12	One or More Eligible [%]	12,08
Issued Warp Per Scheduler	0,12		

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	16,22	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	16,23	Avg. Not Predicated Off Threads Per Warp	31,61

Instruction Statistics

Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

Executed Instructions [inst]	15e032	Avg. Executed Instructions Per Scheduler [inst]	55,26
Issued Instructions [inst]	15e040	Avg. Issued Instructions Per Scheduler [inst]	55,29

NVLink

High-level summary of NVLink connection status.

Physical Links	0	Logical Links	0

Launch Statistics

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	1	Registers Per Thread [register/thread]	25
Block Size	256	Static Shared Memory Per Block [kbyte/block]	2,05
Threads [thread]	256	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	0,00	Driver Shared Memory Per Block [byte/block]	0
Function Cache Configuration	cudaFuncCachePreferNone	Shared Memory Configuration Size [kbyte]	32,77

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	100	Block Limit Registers [block]	8
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	32
Achieved Occupancy [%]	24,99	Block Limit Warps [block]	4
Achieved Active Warps Per SM [warp]	8,00	Block Limit SM [block]	16

Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Sampling Data metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.

Branch Instructions [inst]	784	Branch Efficiency [%]	100
Branch Instructions Ratio	0,05	Avg. Divergent Branches	0

5. Discutez de la vitesse d'exécution du kernel versus les ressources utilisées (shared memory, registres, etc.).

Je constante que lorsque le taille de tuiles augmente, plus que la fonction kernel a un temps d'exécution plus rapide, sauf lorsque le nombre de tuile équivaut à la taille de la matrice. De plus, on peut voir qu'on utilise de plus en plus la mémoire partagé lorsqu'on augmente la taille des tuiles. Cela est normal, car plus que la grosseur des tuiles est élevé, plus que nous devons charger les données dans la mémoire partagé depuis la mémoire globale, ce qui est diminue le nombre d'accès à la mémoire globale et augmente la rapidité d'exécution. Le nombre de registre par thread est constant de 25. Il est évident par contre, que ces implémentations avec plus de tuiles utilise plus de ressources, alors il est important de bien connaitre le GPU avant d'exploiter toutes les ressources pour une vraie application.

6. Comparez et discutez des résultats par rapport à ceux obtenus à la PARTIE 1.

La méthode dans la partie 2 va vraiment plus vite que les méthodes dans la partie 1. La partie 1 fonctionne quand même sauf que celle-ci ne profite pas de la puissance d'un GPU. En effet, l'accès à la mémoire globale peut grandement ralentir la rapidité d'une application. Il est important de bien connaitre les différents types d'accès mémoire et leur latence pour pouvoir créer des applications optimales.

REFERENCES

[2] D. Kirk and W.-M. Hwu, *Programming Massively Parallel Processors- A Hands-on Approach*, 3rd Edition, Morgan Kaufmann, 2017. (Elsevier Science & Technology, ISBN: 978-0-12-811986-0).