Analyse TP GPU

Contexte

Voici une check liste, quelques idées et quelques pistes non exhaustives pour vous guider dans vos TP d'approfondissements.

Check list

Performances

(C1) Utilisation *SM* possible ?

(C2) Utilisation *CM* possible ?

(C3) Elimination des *banks conflits* en *SM*.

Tips :

(T1) Object en SM :

Padding avec un tableau à dimension variable comme attribut.
Faites varier la dimension progressivement.

(T2) De-Objeification :

Stockage des attributs dans des tableaux de type simples.
Travailler avec des données de 4 octets (*float*, *uchar4*, …) dans un u plusieurs tableaux, pour que les 32 threads d'un *warp* tapent dans les 32 bancs de la sm sans conflit.

(C4) *Tiling* des images en *SM*

(C5) Minimisation des transactions mémoires en *GM* (par regroupement en paquets)

Tips :

Variation du pattern de paralélisation :

(V1) *Entrelacement*
(V2) *Un à un*
(V3) *Désentrelacer* (moitié-moitié)

(C6) Optimisation de *(dg,db)*

Tips

(T1) Vérifier les heuristiques
(T2) O*ccupancyCalculator.xls*
(T3) Force brute

Attention :

(A1) A tuner pour chaque *kernel*.
(A2) A *retuner* selon l'utilisation de la *SM* ou non.
(A3) A *retuner* si vous changer de GPU

Note

L'optimum dépend il de la taille des images pour les TP graphiques ?

(C7) Taux occupation empirique vs taux occupation théorique

(C8) Diminution des threads divergence

(C9) Utilisation des textures

(C10) *MultiGPU* possible ?

(C11) P2P enable ?

(C12) Code CPU :

OpenMP : Design parallélisme :

- Entrelacement
- *for-auto*

Compilateur

- GCC
- IntelLinux

(C13) Avez-vous spécifier des flags de compilation CPU et GPU pour optimiser vos codes ?

${ROOT_WORKSPACE}/BUILDER/makefile/**public**/cpp/gcc.mk
${ROOT_WORKSPACE}/BUILDER/makefile/**public**/cuda/cudaGCC.mk

Speed-up

Intro

Le **speedup** est plus intéressant qu'une durée. Pour tout speed-up il faut un référentiel. Le référentiel sera toujours dans un premier temps un code CPU séquentiel.

Exemple

Le code séquentiel a le coefficient **1**. Combien de temps gagne-t-on avec :

- OpenMP ?
- Cuda ?

Note

Ce principe de référentiel s'applique quelques soit l'unité de mesure utilisé :

- **FPS**
- **GB/s**

(S1) Référentiel

Il faut un référentiel :

- CPU mono thread au début (GCC)
- Cuda naif ensuite peut-être

(S2) Unité

Il faut une unité :

- Fps
- GB/s
- …

(S3) Echelle

Il faut une échelle :

- Normal
- Logarithmique

(S4) Evolution

Montrer l'évolution du *speed-up* en fonction des améliorations que vous apportez à vos codes.

(S5) Pertinence

Pour les TP graphiques, utiliser la version sans rendu *OpenGL*.
Les drivers des gpus ont bien été loadé avant de lancer les Chronos ?

```
// Mono GPU
Device::loadCudaDriver(deviceId);

// MultiGPU
Device::loadCudaDriverAll();
```

(S6) Variation du GPU

Utiliser les serveurs cuda1, cuda2, cuda3 pour montrer l'effet du changement de GPU. Le cas échéant re-tuner dg,db, sm, … en focntion du GPU. N'oubliez pas de changer le flag de compilation pour *targetter* au mieux le GPU.

(S7) Variation du compilateur CPU

Linux :

- gcc
- intel

Windows :

- Visual
- Mingw
- Intel

(S8) Piste

(P1) Compter le nombre de GB/s que l'on obtient en copiant des données de GM à GM (avec et sans p2p). Comparer ensuite avec le GB/s de vos codes et quantifier ce que « coute » vos calculs. On ne peut pas faire mieux en GB/s qu'une simple « copie ».

(P2) Quelles performances obtient-on avec un core Cuda contre un core cpu ? Combien de MP doit on avoir au minimum pour concurrencer un core CPU.

(S9) Interprétation

Attention, les serveurs *cuda* contiennent respectivement :

- Cuda1 : 12 cores
- Cuda2 : 16 cores
- Cuda3 : 32 cores

avec le multithreading désactivé ! A tenir en compte lors de vos interprétations !

(S10) Collaboration

En cas de *force brute*, avertissez vos collègues par mail pour réserver :

- une tranche horaire
- un ou plusieurs gpu

Spécifier le serveur (Cuda1, Cuda2, Cuda3)

(S11) Control

Avant et surtout après vos *speed-up*, contrôler l'état du serveur avec :

- *nvidia-smi --loop=1*
  (mais ne le laisser pas tourner trop, consomme des ressources !)

- *htop*

Analyse

Attention, n'oubliez pas de confronter vos résultats avec la théorie, et analyser les !
Vos résultats empiriques sont-ils en adéquation avec vos prévisions théoriques ?

Exemple

Taux d'occupation théorique du GPU versus Taux d'occupation empirique

Compléxité

Mesurer les dégradations des performances de votre implémentation.

Par exemple si on double la taille de l'image. Le temps double-t-il aussi? La **complexité** de votre implémentation est :

- Linéaire ?
- Quadratique ?
- …

Que peut-on dire des speed up ? Ils sont invariants à la taille des images ?

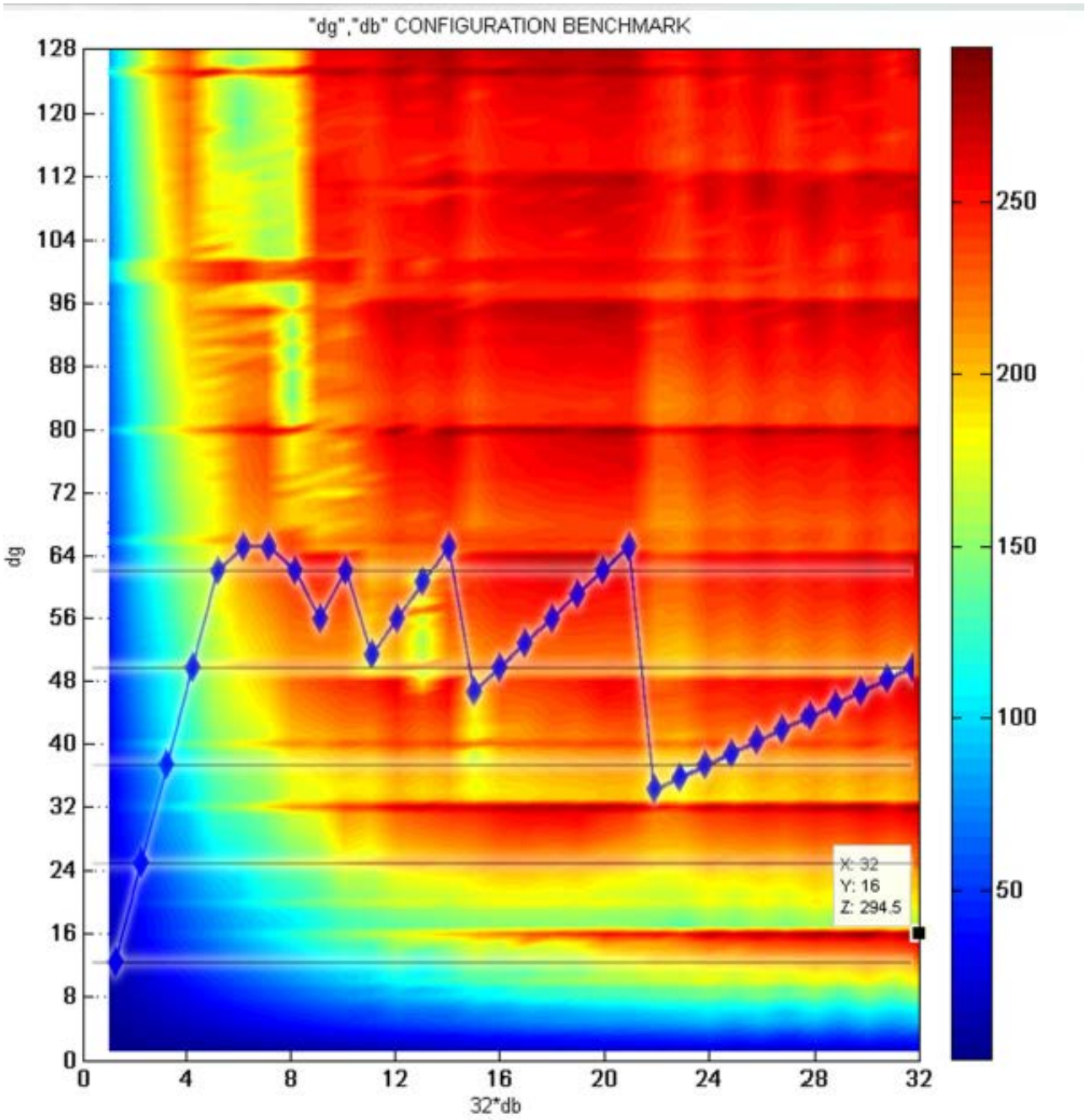
Graphiques

Illustrer vos résultats avec des graphiques :

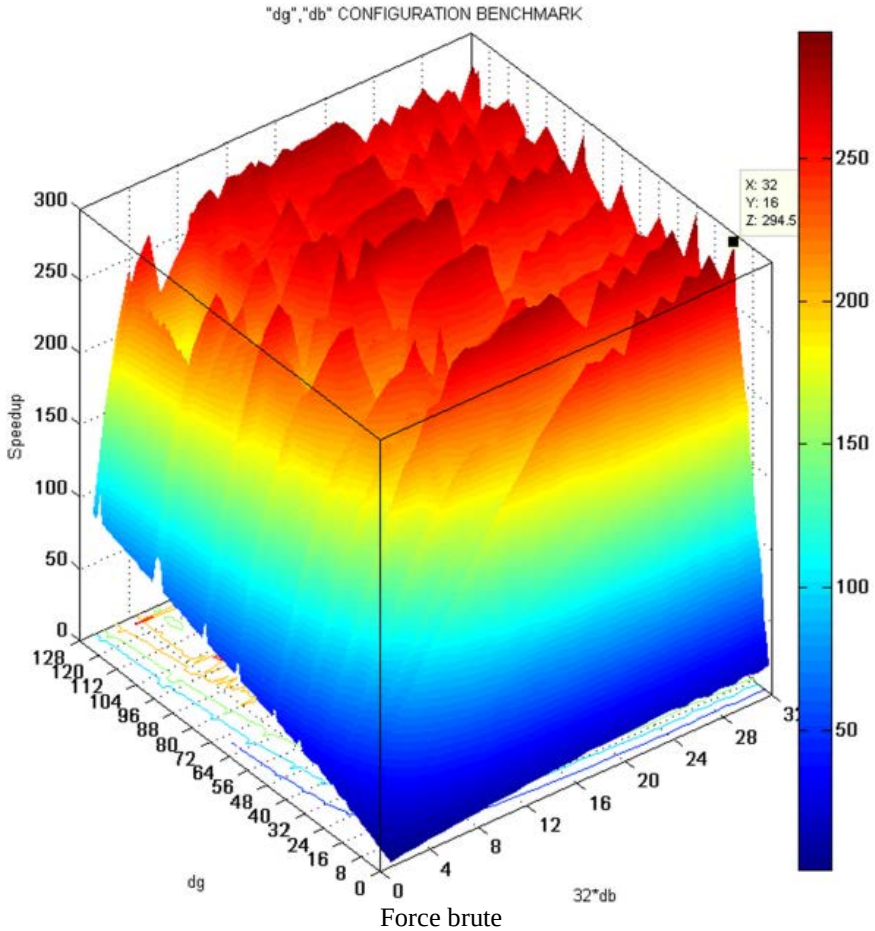
- Courbes 1D
- Surfaces 2D
- Courbes de niveau
- Camembert
- …

Soyez imaginatif !
Le cas échéant utiliser une échelle logarithmique !
N'oubliez pas d'interpréter vos graphiques !

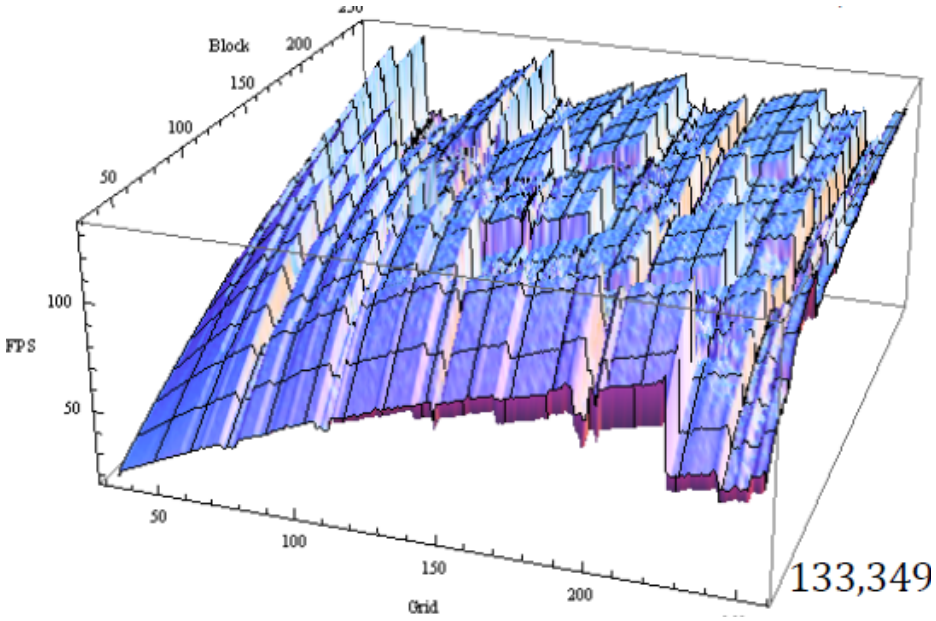
Exemples



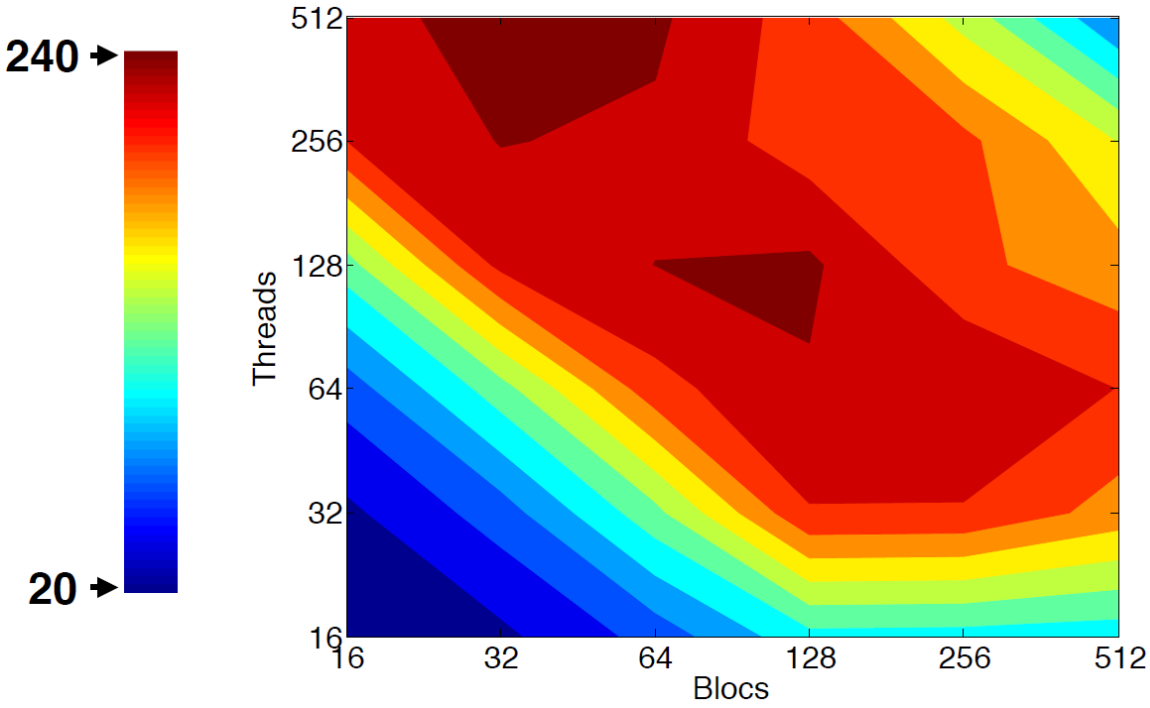
Force brute / comparaison théorie : **EXCELLENT** !



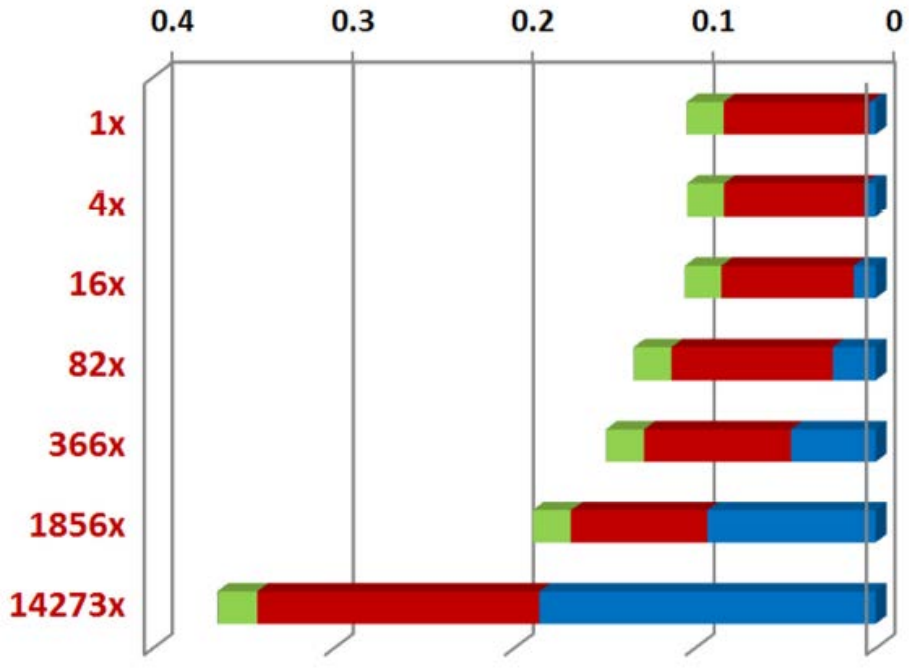
Force brute



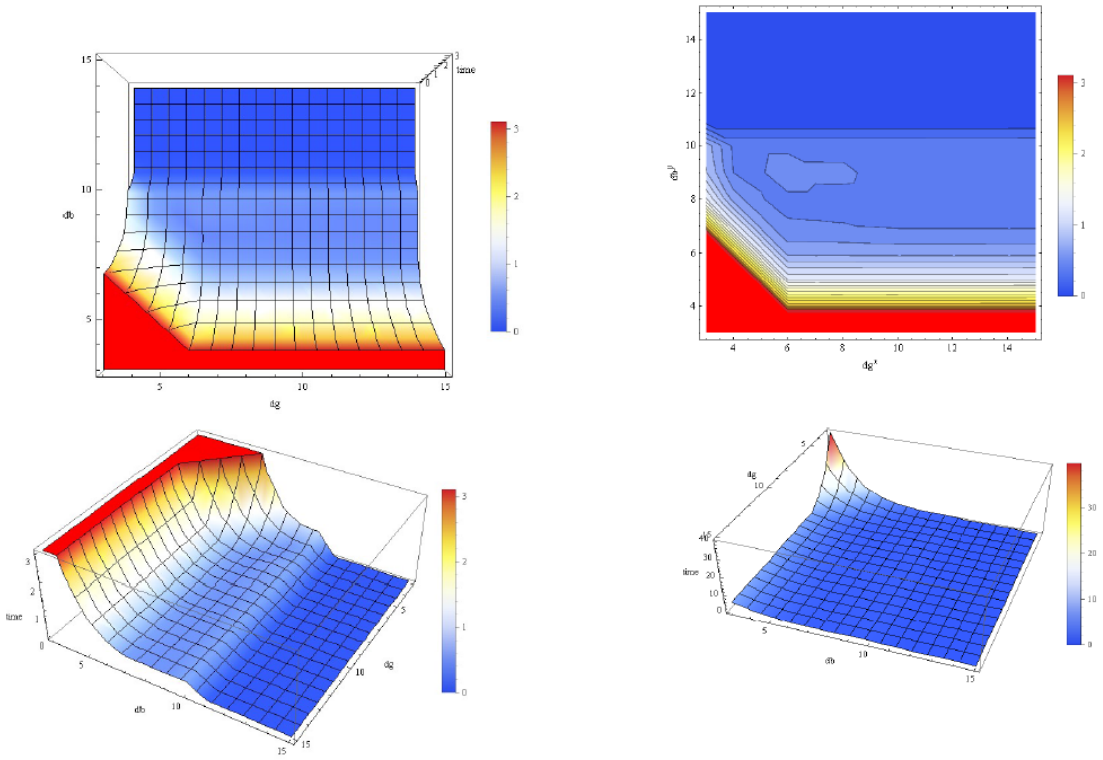
Force brute, à intrepréter correctement !



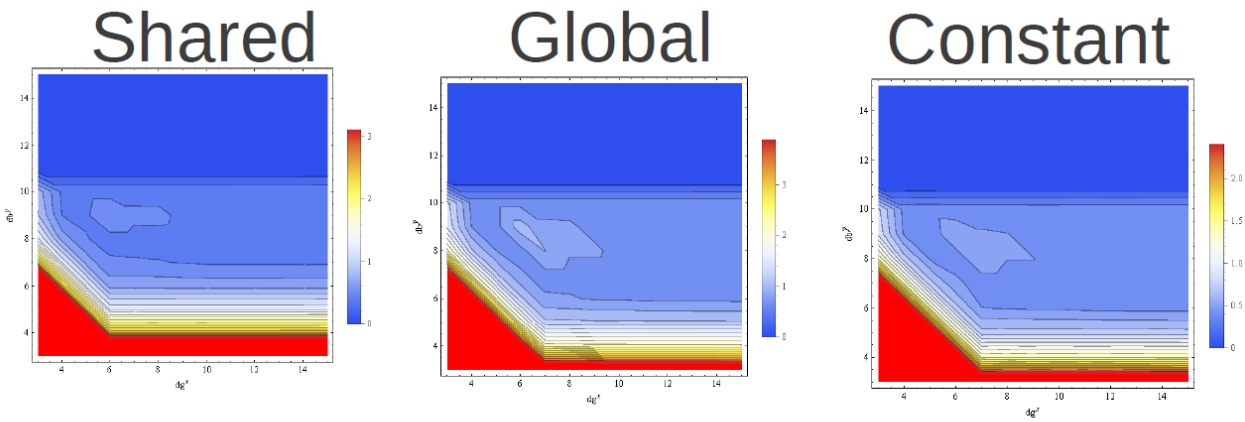
Force brute

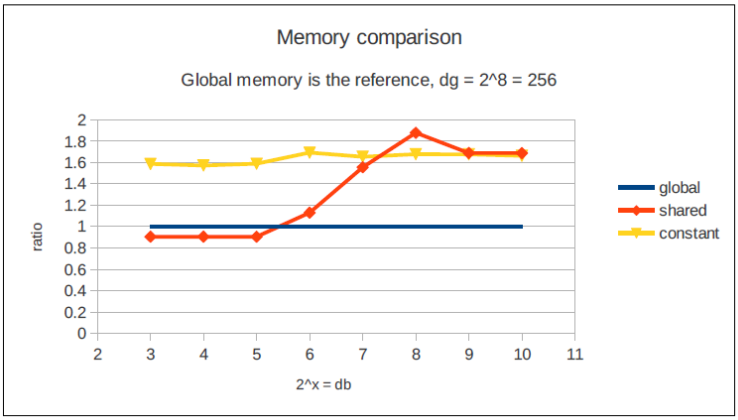


Pourcentage entre activités

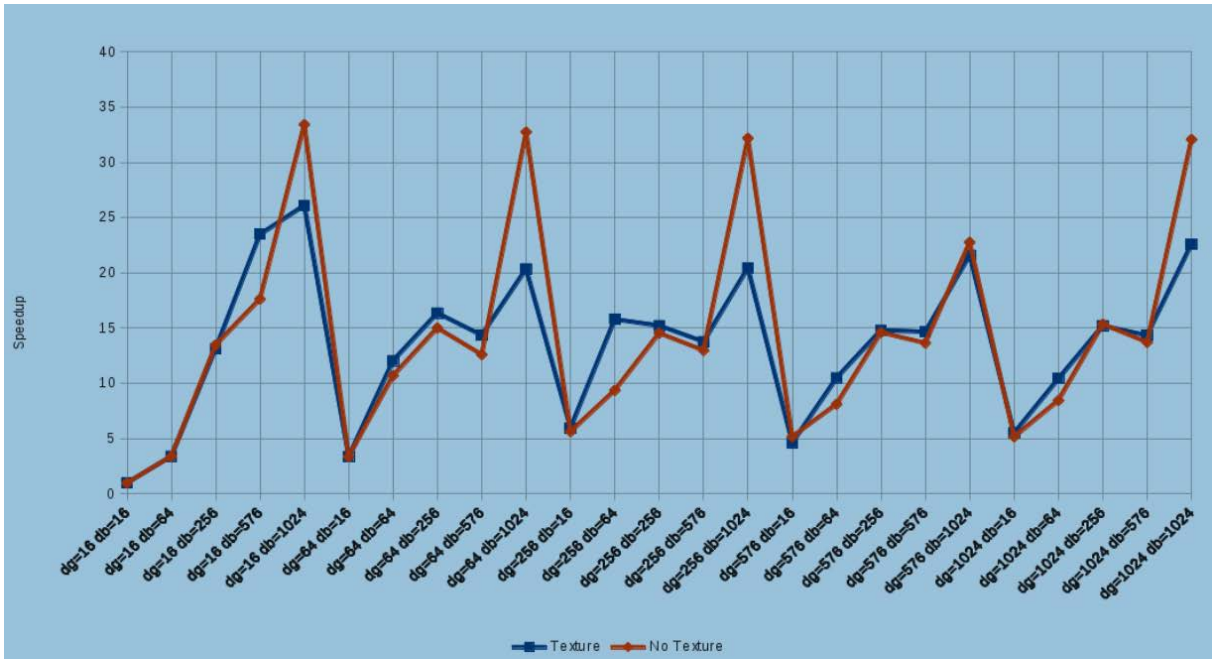


Force brute





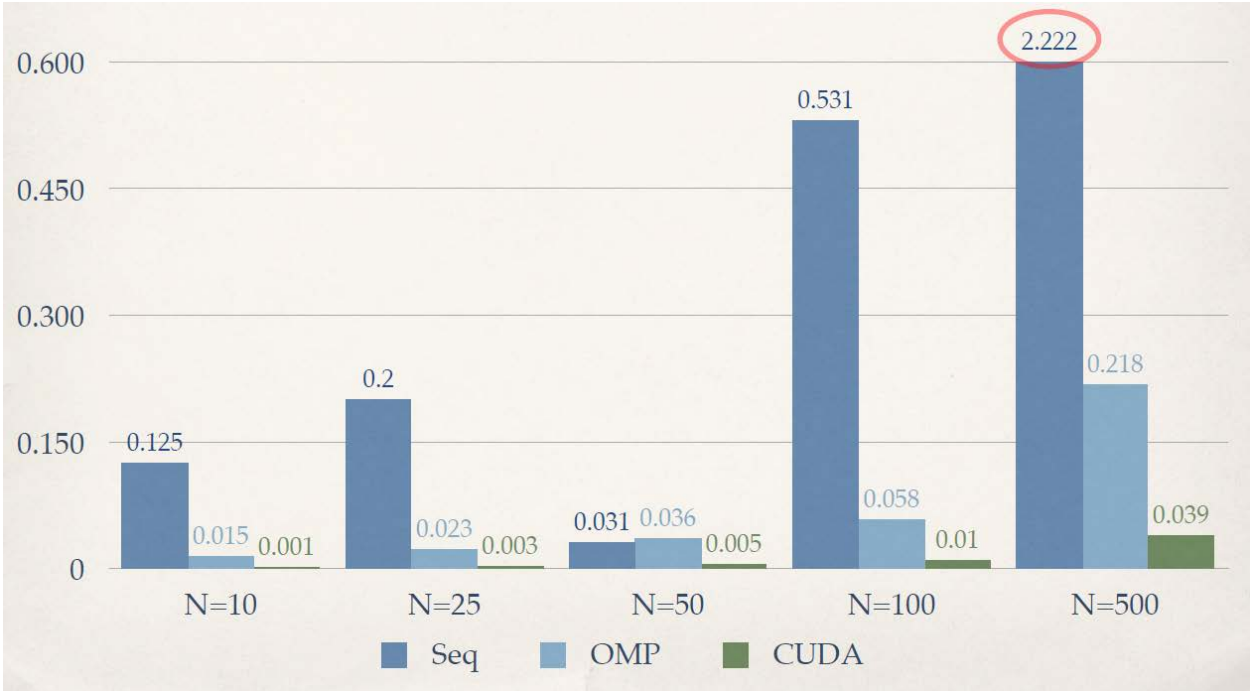
Force brute


Speedup
0
5
10
15
20
25
30
35
40
dg=16 db=16
dg=16 db=64
dg=16 db=256
dg=16 db=576
dg=16 db=1024
dg=64 db=16
dg=64 db=64
dg=64 db=256
dg=64 db=576
dg=64 db=1024
dg=256 db=16
dg=256 db=64
dg=256 db=256
dg=256 db=576
dg=256 db=1024
dg=576 db=16
dg=576 db=64
dg=576 db=256
dg=576 db=576
dg=576 db=1024
dg=1024 db=16
dg=1024 db=64
dg=1024 db=256
dg=1024 db=576
dg=1024 db=1024
Texture
No Texture

Force brute


1.500
1.125
0.750
0.375
0
1.331
0.708
0.134
0.075
0.018
Seq GCC
Seq Intel
OMP GCC
OMP Intel
CUDA


30
22.5
15
7.5
0
N=10
N=25
N=50
N=100
N=500
6.96 x
8.37 x
11.96 x
8.17 x
17.58 x
8.40 x
21.80 x
9.50 x
23.15 x
10.29
OMP
CUDA


0.600
0.450
0.300
0.150
0
N=10
N=25
N=50
N=100
N=500
0.125
0.015
0.001
0.2
0.023
0.003
0.031
0.036
0.005
0.531
0.058
0.01
2.222
0.218
0.039
Seq
OMP
CUDA

Linux		Relative speedup		Absolute speedup
	Time	Same compiler	Previous version	
Seq GCC	0.504 s	–	–	1.00 x
Seq Intel	0.247 s	–	2.04 x	2.04 x
OMP GCC	0.026 s	19.06 x	9.35 x	19.06 x
OMP Intel	0.012 s	19.99 x	2.14 x	40.77 x
CUDA	0.001 s	39.39 x	18.42 x	750.92 x

End