
LNS sur architectures exotiques

Mark Arnold, Sylvain Collange et David Defour

Lyon, 27 octobre 2009



Objectifs

- Évaluer des fonctions univariées sur GPU
- Exploiter les unités de filtrage
- Application au système logarithmique (LNS)

Plan

- Filtrage de texture
- Évaluation de fonction
- Système logarithmique (LNS)
- Résultats

Types d'unités sur GPU

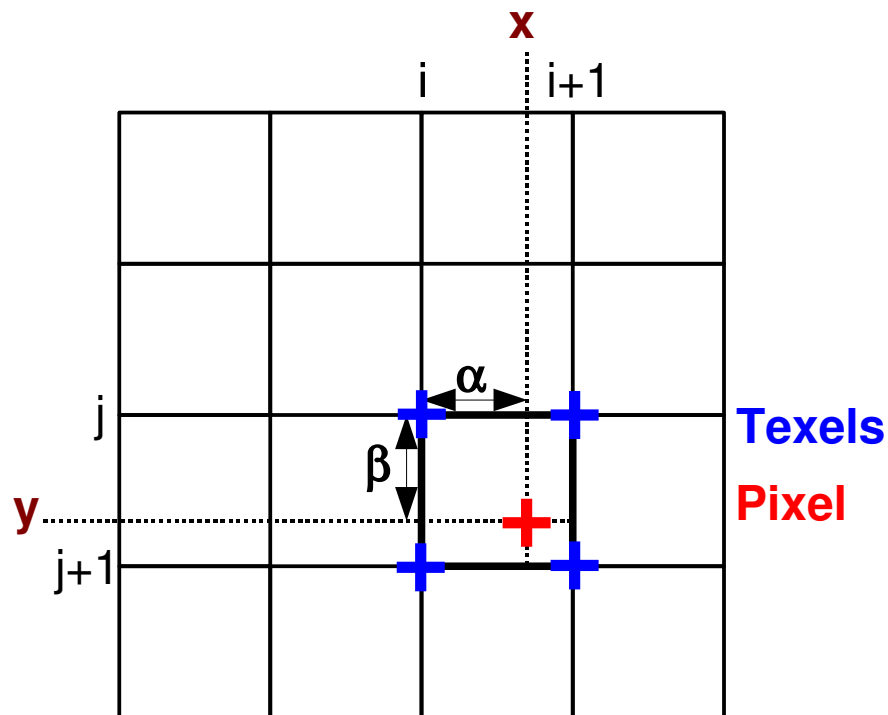
- Plusieurs types d'unités
 - ◆ Pouvant fonctionner en parallèle

Type	Utilisation	Facteur limitant
MAD	Calcul généraliste	Calcul, registres
SFU	Fonctions élémentaires	Calcul
Interpolateur	Graphique	Calculs d'adresse
Texture	Graphique	Bande passante
Autres	...	

- En GPGPU, seuls les unités de calcul de base sont exploitées
- Tirer partie des autres unités?
 - ◆ Texture

Filtrage de texture

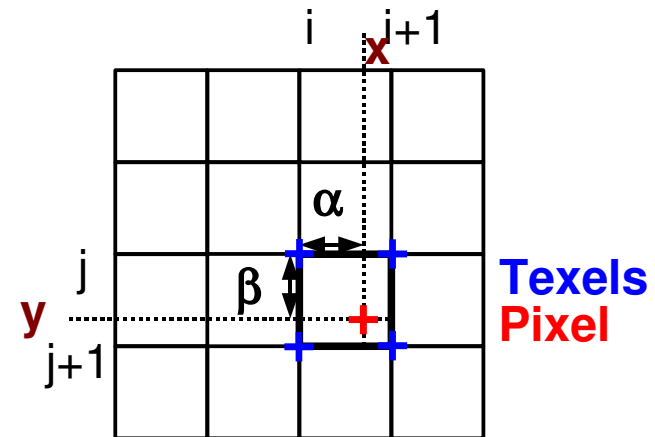
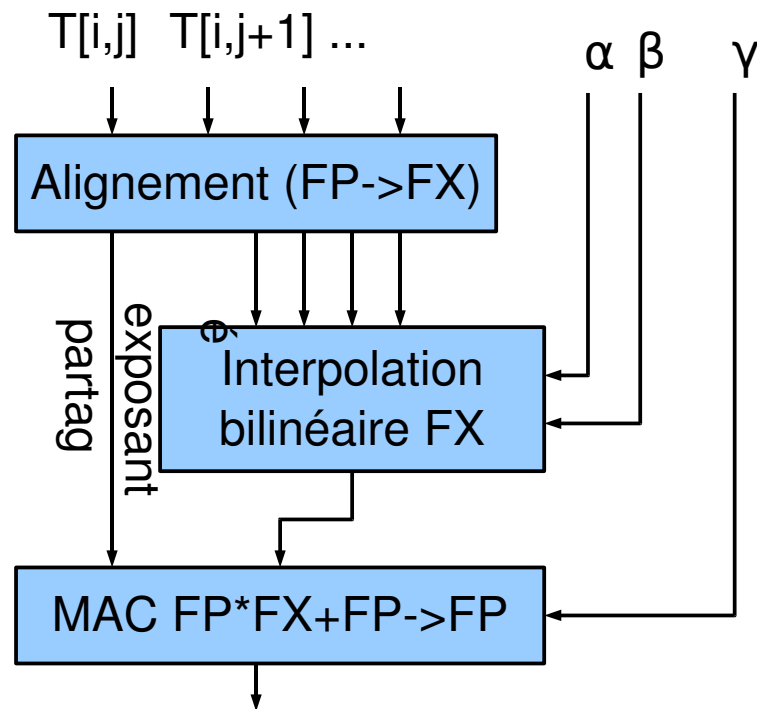
- Pour chaque **pixel** (x,y) de l'image
 - ◆ Interpolation à partir des **texels** les plus proches
 - ◆ $i = \text{floor}(x)$, $\alpha = \text{frac}(x)$, $j = \text{floor}(y)$, $\beta = \text{frac}(y)$



$$\begin{aligned} \text{pix}(x, y) = & (1-\alpha) (1-\beta) T [i, j] + \alpha (1-\beta) T [i+1, j] \\ & + (1-\alpha) \beta T [i, j+1] + \alpha \beta T [i+1, j+1] \end{aligned}$$

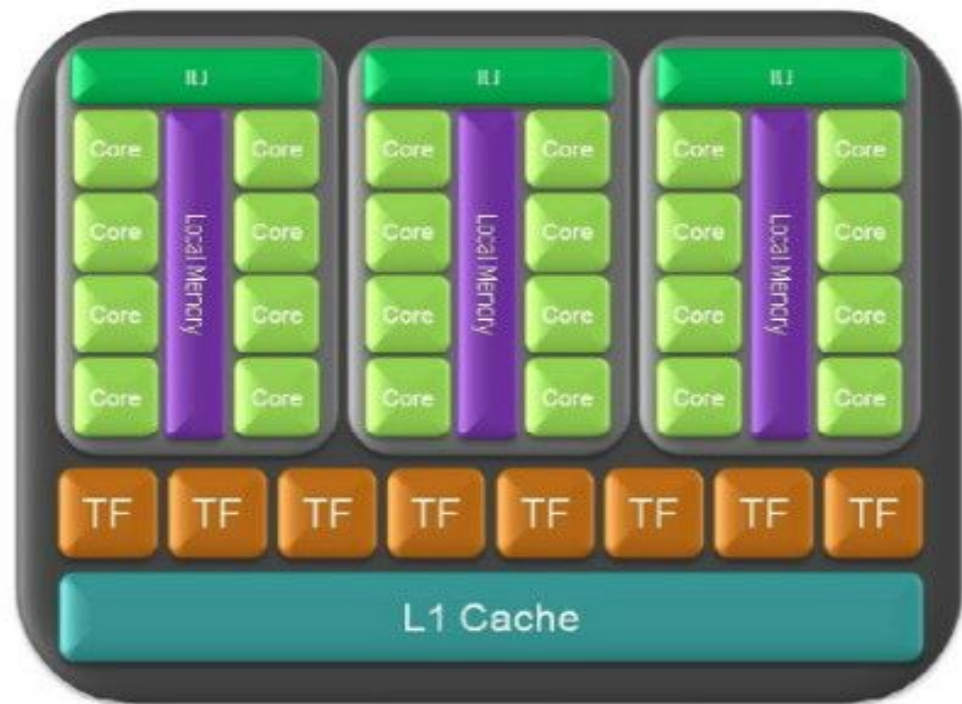
Unités de filtrage

- Unité de calcul spécialisée (ex NVIDIA GT200)
- $T[i,j]$ en virgule flottante simple précision
- α , β en virgule fixe, 8 bits fractionnaires
- Largeur des chemins de données internes?



Placement, performance

- Filtrage bilinéaire « gratuit » sur le chemin vers la mémoire
- En sortie du cache de texture (8 Ko/cœur)



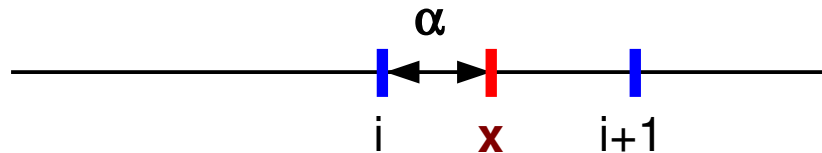
GTexels/s	GTS 250	GTX 285	HD 4890	HD 5770	HD 5870
32 bits - 4x INT8	40.0	52.2	27.0	33.8	66.7
32 bits - RGB9E5	23.5	25.8	27.3	33.8	68.6
64 bits - 4x FP16	23.5	26.1	17.0	17.0	33.8
128 bits - 4x FP32	11.8	12.9	8.5	8.5	16.9

Plan

- Filtrage de texture
- Évaluation de fonction
- Système logarithmique (LNS)
- Résultats

Cas 1D

- Filtrage linéaire



$$\text{pix}(x) = (1-\alpha)T[i] + \alpha T[i+1]$$

- Permet d'approximer des fonctions univariées
- Cas particulier du filtrage bilinéaire
 - ◆ Matériel partiellement exploité
- Utiliser la seconde dimension pour augmenter la précision?

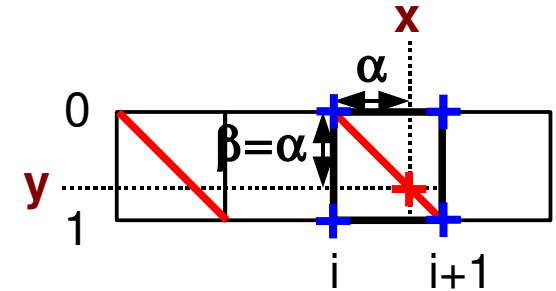
Degré 2

- $\beta = \alpha, j=0$

$$\left\{ \begin{array}{l} \text{pix}(x, y) = (1-\alpha)^2 T[i,0] + \alpha(1-\alpha) T[i,1] + \alpha(1-\alpha) T[i+1,0] + \alpha^2 T[i+1,1] \\ = a_2 \alpha^2 + a_1 \alpha + a_0 \end{array} \right.$$

Polynôme de degré 2 en α

$$\left\{ \begin{array}{l} T[i,0] = a_0 \\ T[i+1,0] + T[i,1] = a_0 + a_1 \\ T[i+1,1] = a_0 + a_1 + a_2 \end{array} \right.$$



- Texture $2n \times 2$

- $y' = \text{frac}(x), x' = 2 \times \text{floor}(x) + \text{frac}(x)$

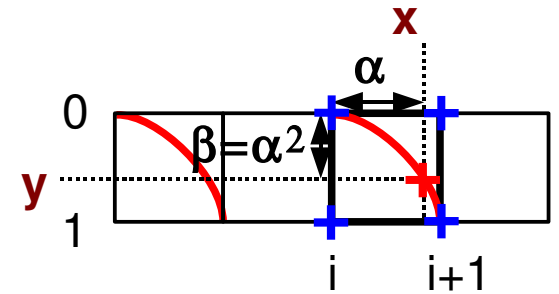
- Filtrage bilinéaire \rightarrow évaluation de polynôme de degré 2

Degré 3

- $\beta = \alpha^2, j=0$

$$\left\{ \begin{array}{l} \text{pix}(x, y) = (1-\alpha) (1-\alpha^2) T[i, 0] + \alpha (1-\alpha^2) T[i+1, 0] \\ \quad + (1-\alpha) \alpha^2 T[i, 1] + \alpha^3 T[i+1, 1] \\ = a_3 \alpha^3 + a_2 \alpha^2 + a_1 \alpha + a_0 \end{array} \right.$$

$$\left\{ \begin{array}{l} T[i, 0] = a_0 \\ T[i+1, 0] = a_0 + a_1 \\ T[i, 1] = a_0 + a_2 \\ T[i+1, 1] = a_0 + a_1 + a_2 + a_3 \end{array} \right.$$



- Carré + filtrage bilinéaire
→ évaluation de polynôme de degré 3

Plan

- Filtrage de texture
- Évaluation de fonction
- **Systeme logarithmique (LNS)**
- Résultats

- Logarithmic Number System
- Système de représentation de réels
- On représente les nombres par leur \log_2 généralement stocké en virgule fixe (e, f)
- Signe s
- Valeurs spéciales : 0, éventuellement $+\infty$, $-\infty$ et NaN

$$m = (-1)^s \cdot 2^{\overline{e, f}}$$

- Marché de niche, sur FPGA
 - ◆ Sur GPU?

Opérations en LNS

- $\times, \div, \sqrt{\quad}$ \rightarrow addition, soustraction, décalage
- $+, -$ \rightarrow évaluation de fonctions

pour $x > y > 0$:

$$L_{x+y} = L_x + \underline{s}_2(L_y - L_x) \quad \text{Somme, signes } \text{semblables}$$

$$L_{x-y} = L_x + \underline{d}_2(L_y - L_x) \quad \text{Différence, signes } \text{différents}$$

avec :

$$s_2(z) = \log_2(1 + 2^z)$$

$$d_2(z) = \log_2(1 - 2^z)$$

Représenter l'exposant

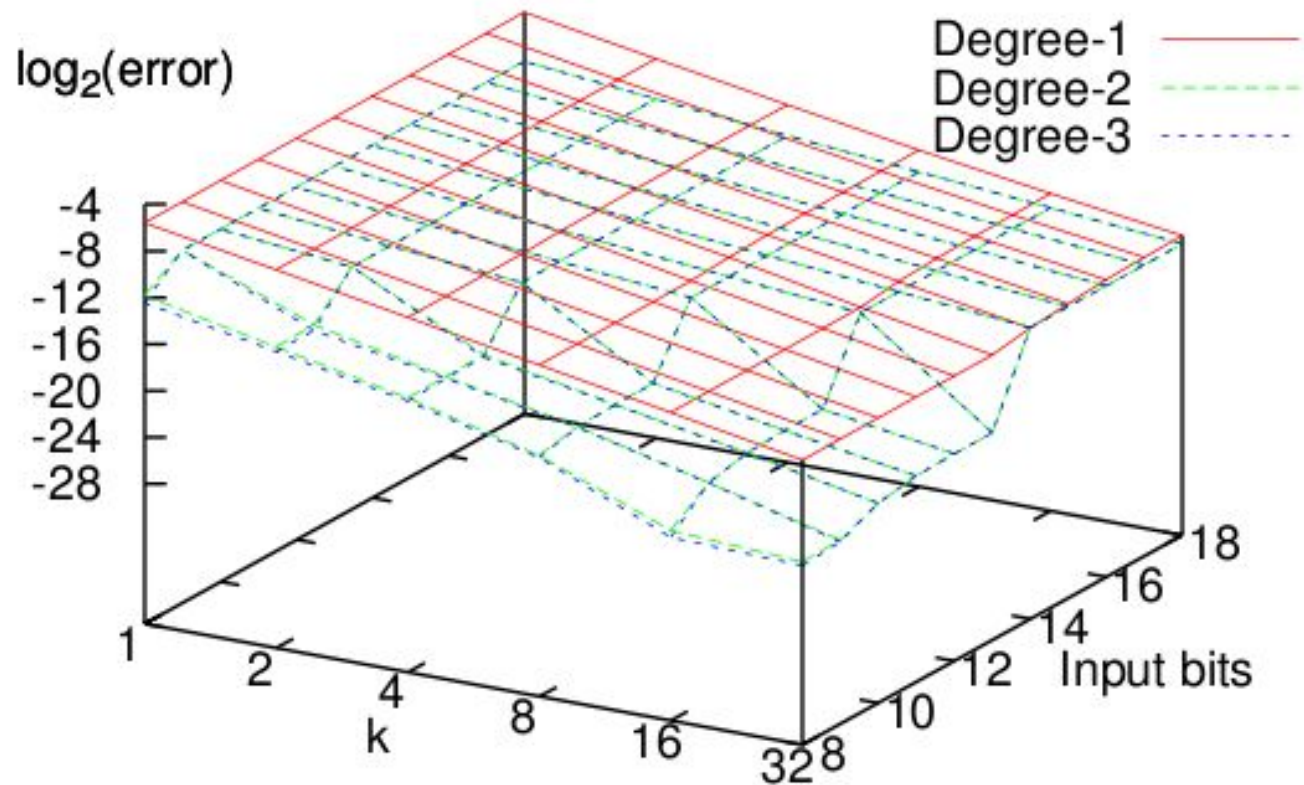
- Typiquement, virgule fixe en complément à 2
- Sur GPU, peu de calcul entier
 - ◆ Exposant en virgule flottante
 - ◆ Dynamique résultante énorme

Plan

- Filtrage de texture
- Évaluation de fonction
- Système logarithmique (LNS)
- Résultats

Précision

- Texture $2k \times 2$
- $s_b(z) = \log_2(1+2^z)$



Efficacité

- Gops/s, sur NVIDIA GeForce 9800 GTX+

Méthode	s_2	s_2 et d_2
Ordre 1	18	18
Ordre 2	16	16
Ordre 3	16	16
Unités SFU	24	23
Horner ordre 5	22	17
Ordre 2 + SFU	17	15
2x SFU	19	14
FloPoCo, XC4VLX200 (est)	25	8

Du LNS sur GPU, quelle drôle d'idée

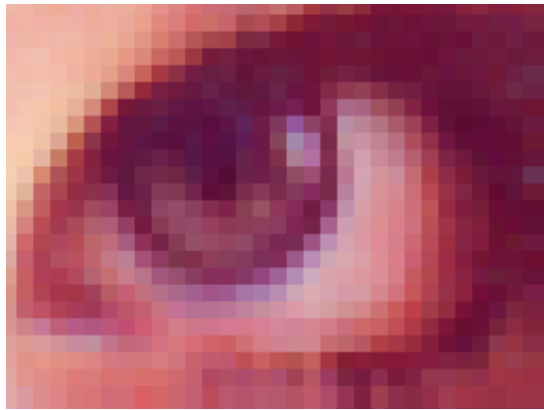
- Caractéristiques du LNS
 - ◆ Système concurrent à la virgule flottante
 - ◆ Transforme multiplications en additions
 - ◆ Utilise des méthodes à base de tables
- Caractéristiques du GPU
 - ◆ Grande puissance de calcul en virgule flottante
 - ◆ Multiplication au moins aussi rapide que l'addition
 - ◆ Unités d'inverse, racine carrée, exp/log pipelinées
 - ◆ Peu de caches, latence élevée, faible débit / flop
- Malgré tout, résultats honorables

Conclusion

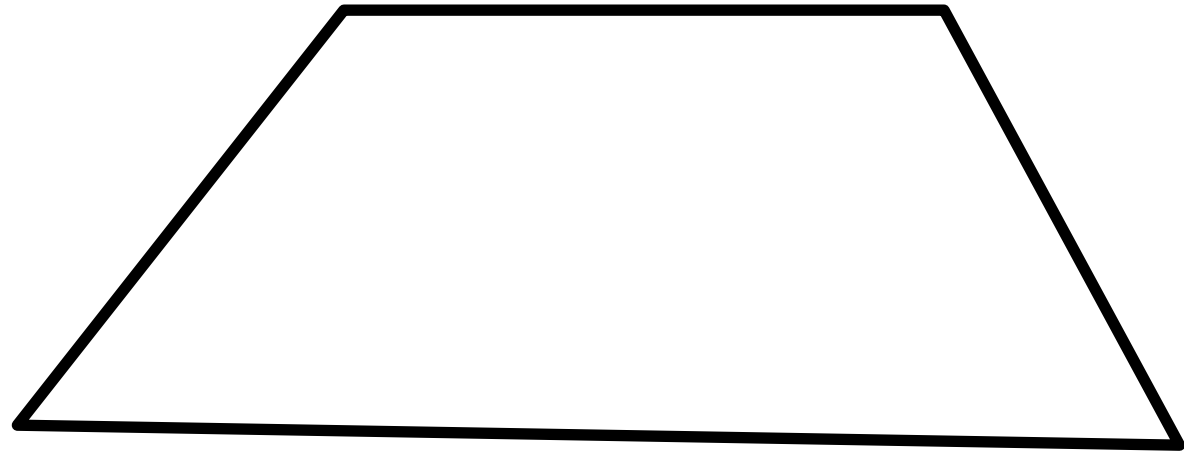
- Possible de détourner une unité graphique
- Moins de 50 fois plus lent qu'un FPGA
- Problèmes de précision, performance relative
- Accès à l'interpolateur d'attribut?
 - ◆ Unité virgule flottante précise
 - ◆ Débit identique à SFU

Filtrage de texture

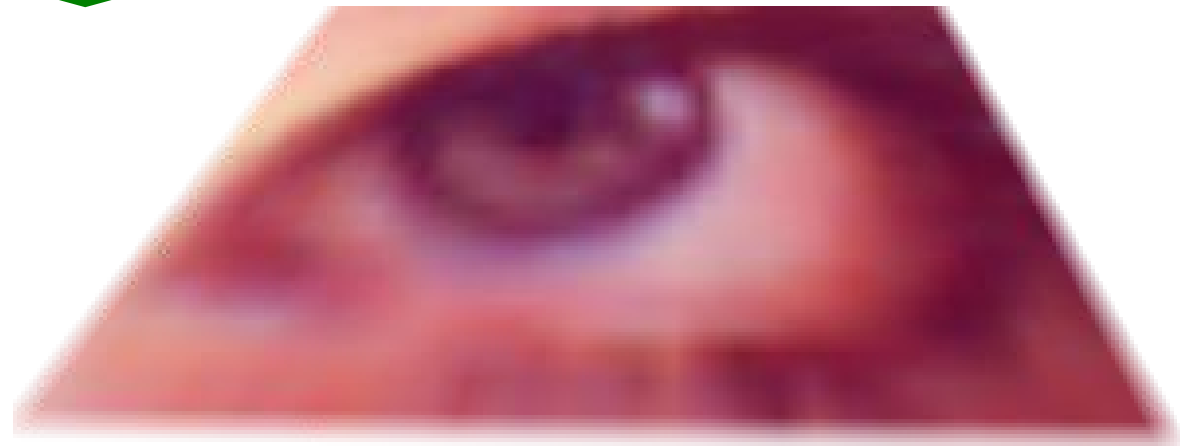
- Interpolation bilinéaire entre texels



Texture



Polygone



Polygone texturé