

Optimisation de la hiérarchie mémoire de traitements non-linéaires dans un flot de synthèse de haut niveau

S. Mancini



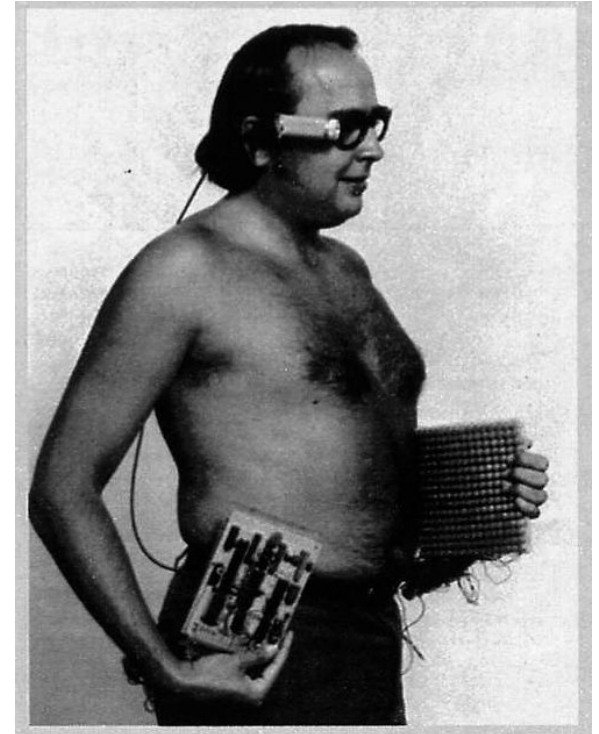
Contexte

Systèmes embarqués pour le traitement d'image :

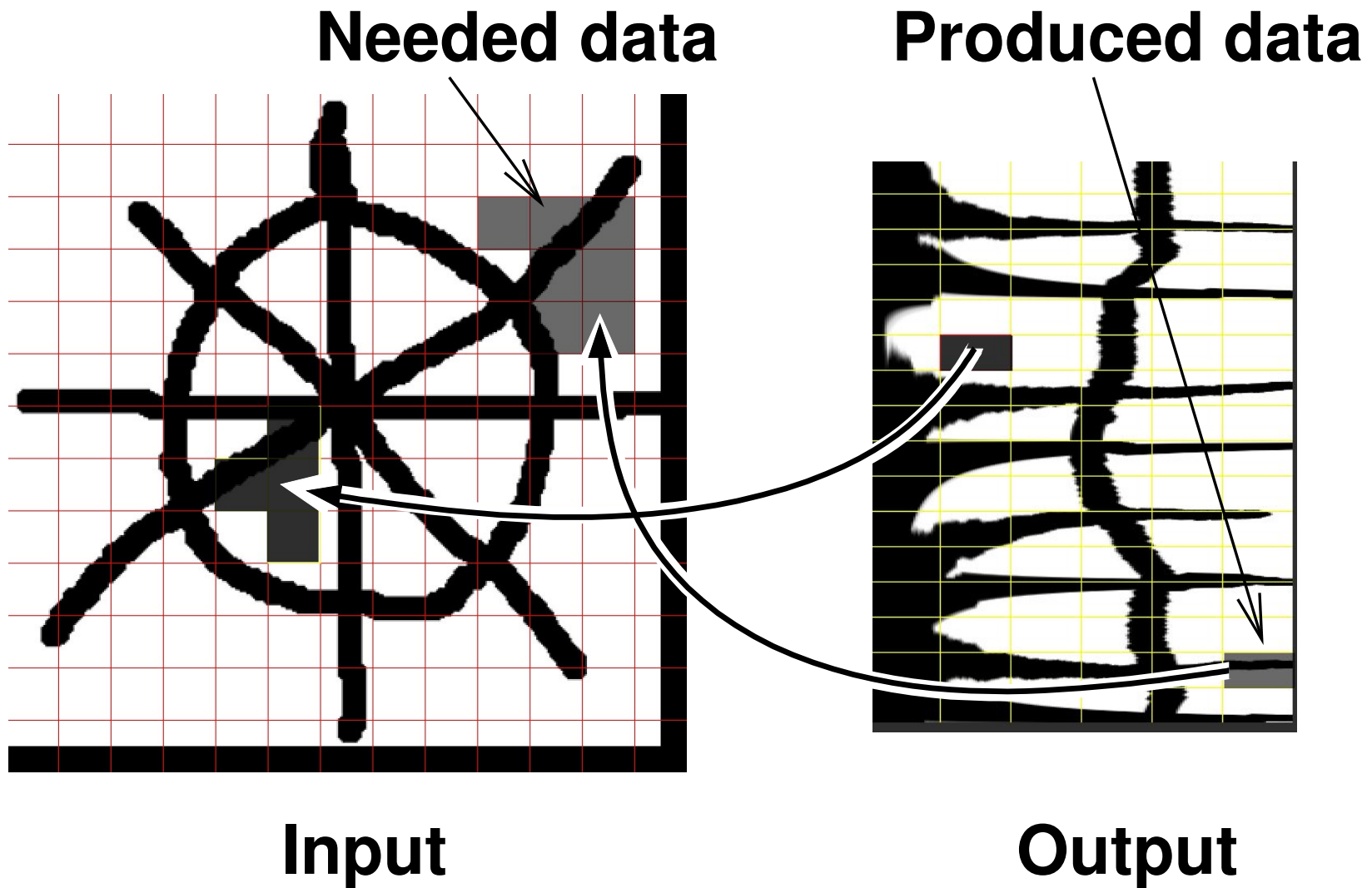
- Traitements spécifiques
 - ☆ Traitements “bas niveau”
 - ☆ Images haute définition
- Contraintes des architectures
 - ☆ Ressources limitées
 - ☆ Basse consommation
 - ☆ Temps réel

La gestion du transport des données entre la mémoire (externe) et les unités de traitement devient le facteur limitant :

➤ Hiérarchie mémoire adaptée



Traitement de lois d'accès non-linéaires



Problématique

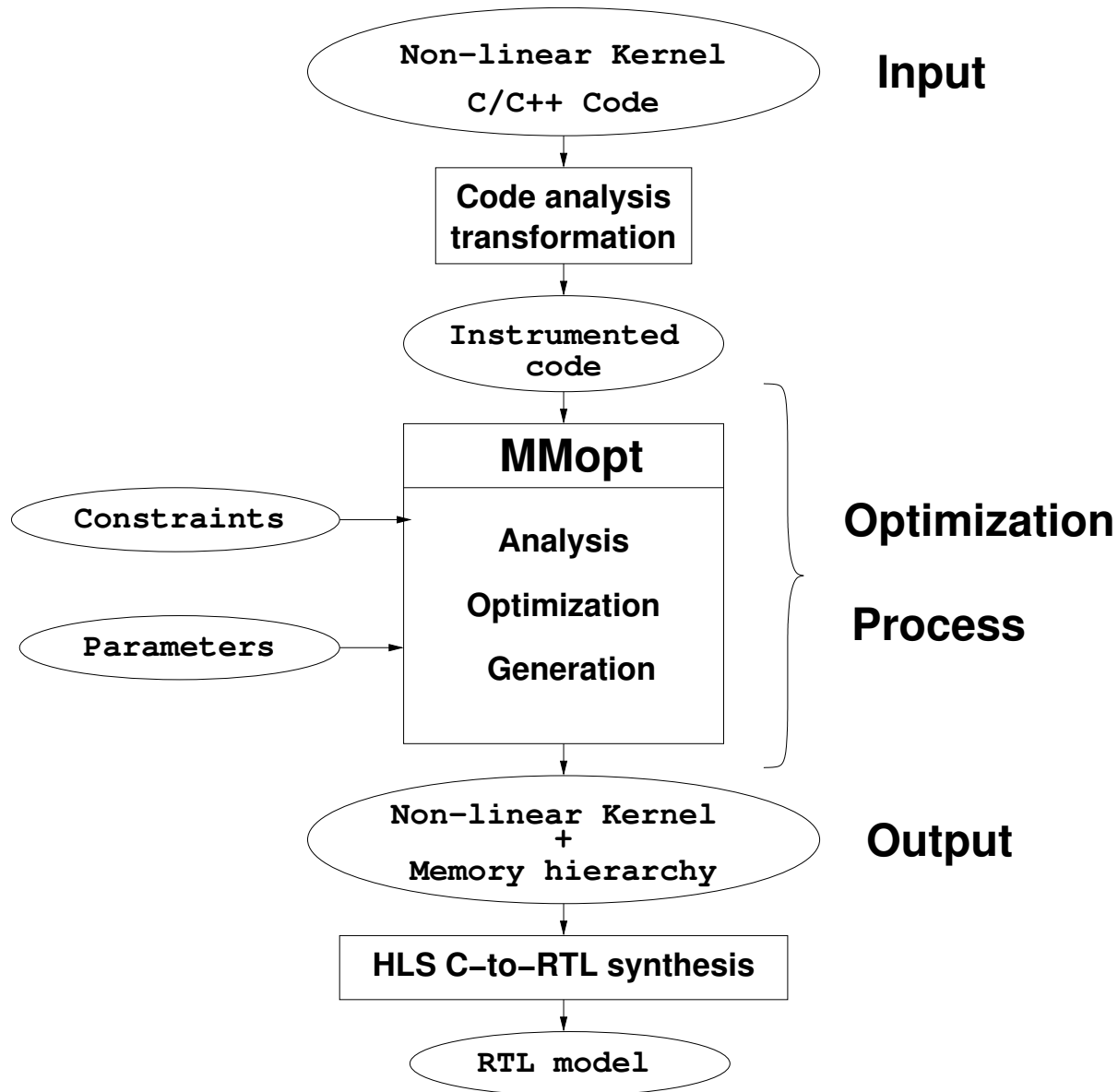
Optimiser la gestion des données et la hiérarchie mémoire dans un flot de HLS

- ? Comment gérer la disparité des lois d'accès non-linéaires
- ? sous la contrainte des outils de HLS ?
- ? les données sont supposées présentes en début de
- ? calcul
- ? Le contrôle doit être très régulier
- ? Les méthodes standards (polyédrique, etc) d'optimisa-
- ? tion des hiérarchie mémoire sont inadaptées

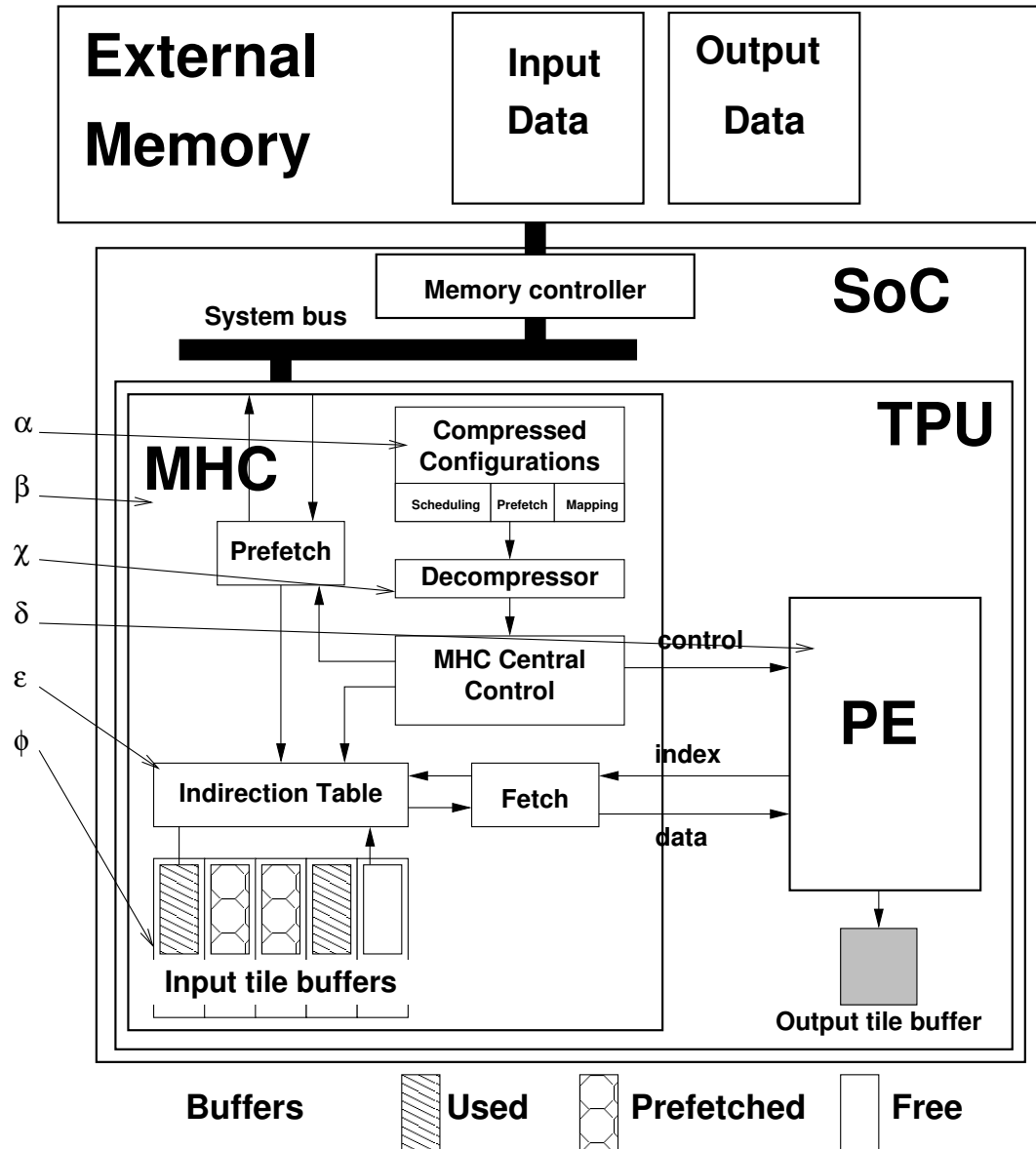
Idées :

- Enrichir un modèle d'entrée pour la HLS par une hiérarchie mémoire
- Générer un modèle "régulier" du point de vue HLS
- Gérer la disparité en "amont"

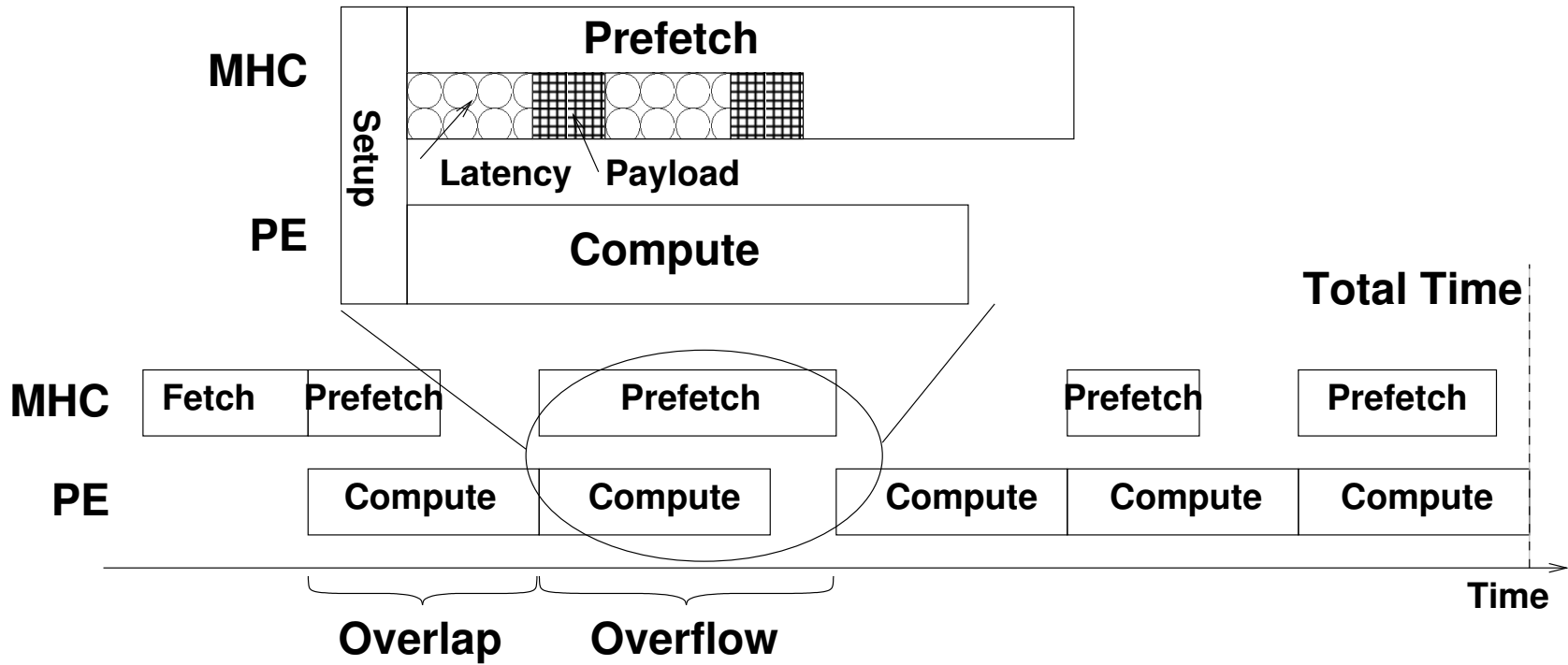
Vue générale du flot



Architecture cible



Séquencement



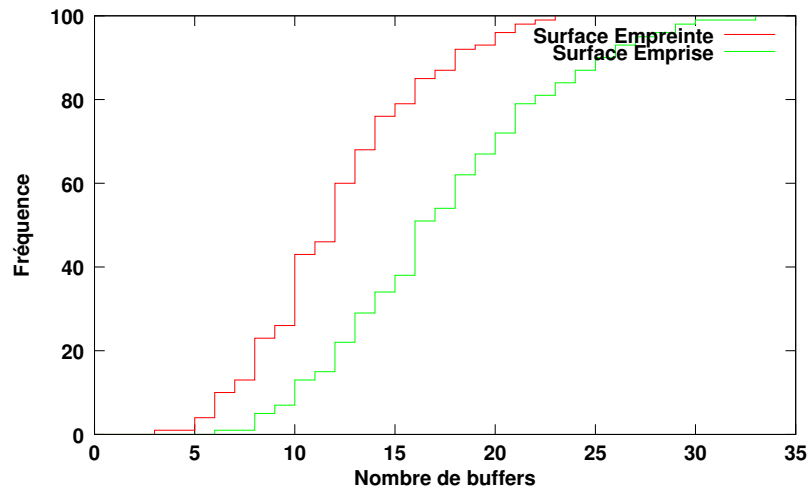
Optimisations

Objectifs :

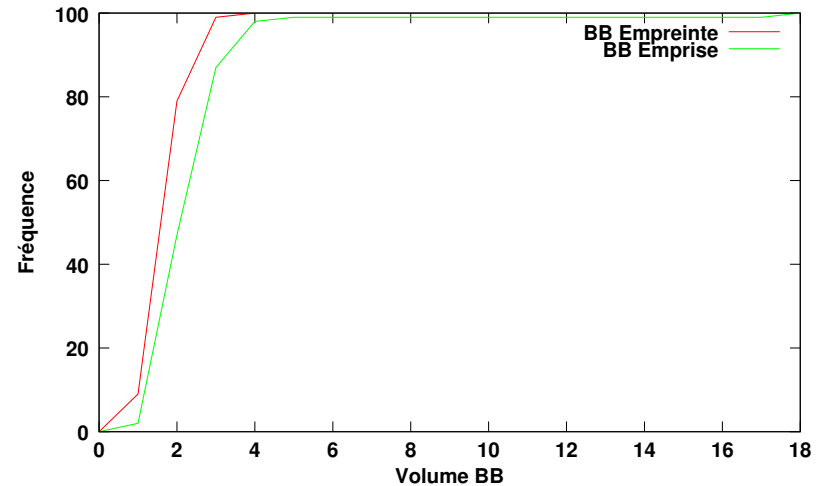
- Minimiser la quantité de données chargées (trafic)
 - Calculer un ordonnancement qui minimise une fonction de coût
- Minimiser la table d'indirection
 - Réduire l'interdistance pour une allocation directe multi-dimensionnelle basée modulo
- Minimiser le nombre de buffers
 - Réduire l'emprise (Hold) maximum (calcul+prefetch)
- Minimiser les mémoires de configuration
 - Scheduling
 - Mapping
 - Pre-fetch
 - Compression RLE

Analyse

Les ressources matérielles sont dimensionnées sur le pire cas qui, en fait, n'est atteint que très rarement.

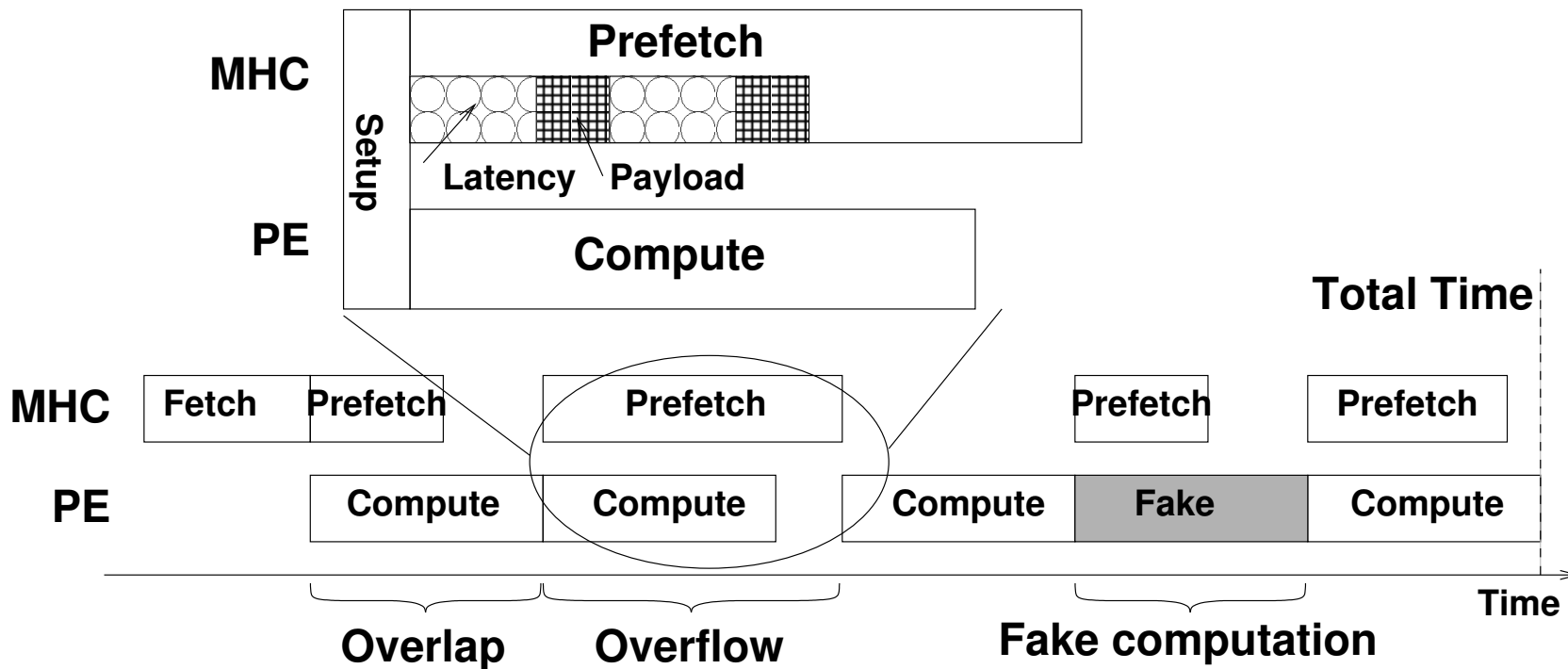


Histogramme du nombre de buffers

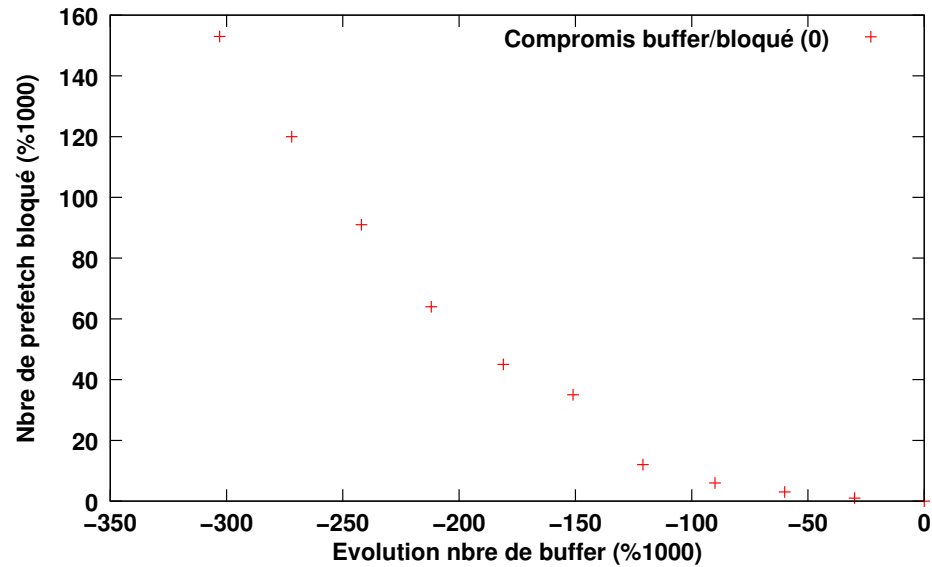


Histogramme du volume de boîte englobante maximum

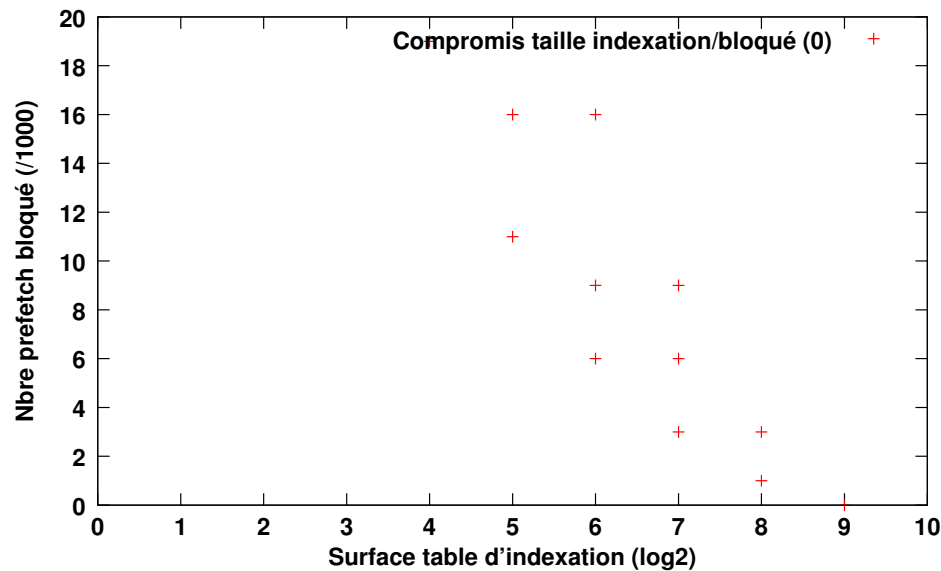
Gestion de la disparité



Compromis disponibles



Compromis nombre de buffers/temps



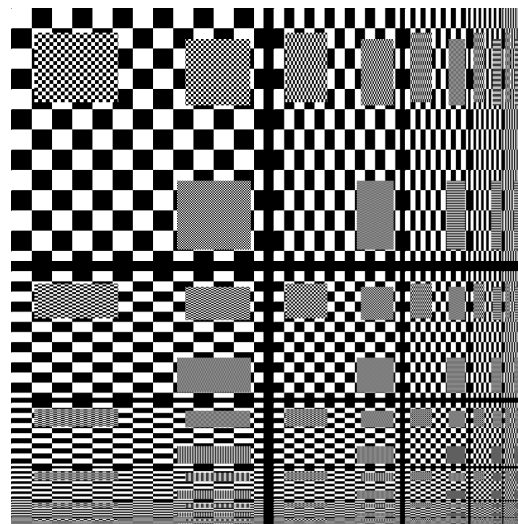
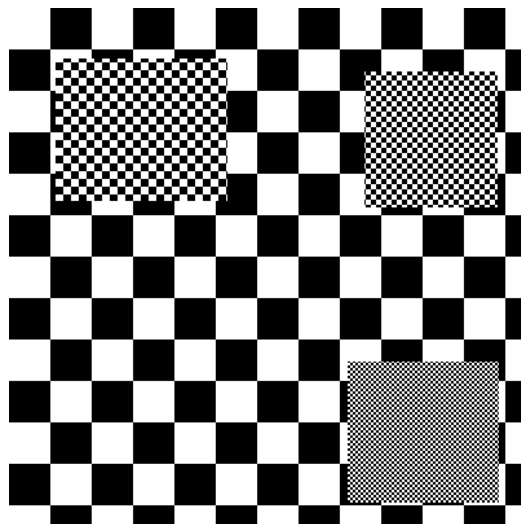
Compromis surface table d'indexation/temps

Benchmarks

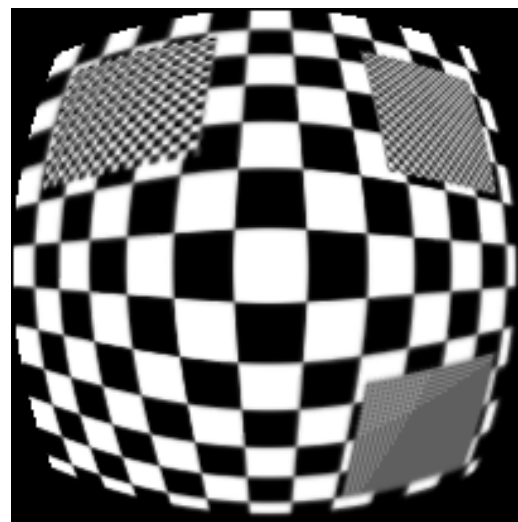
- ★ Traitements :
 - Echantillonnage pseudo-log
 - Transformation polaire
 - Correction grand angle (fisheye)
- ★ Données :
 - Image simple
 - MipMap anisotropique
 - MipMap isotropique

Mipmap ?

Entrée



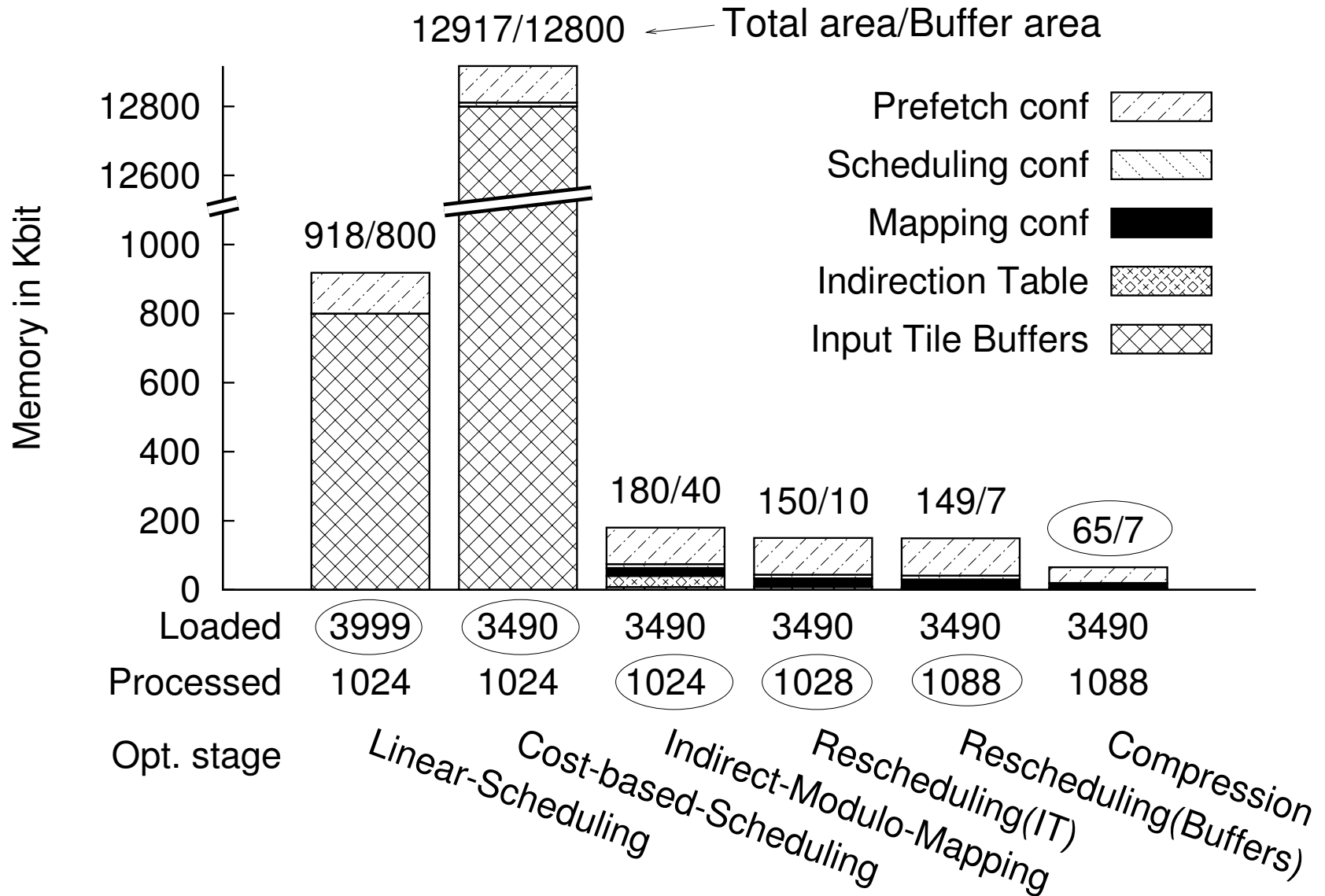
Sortie



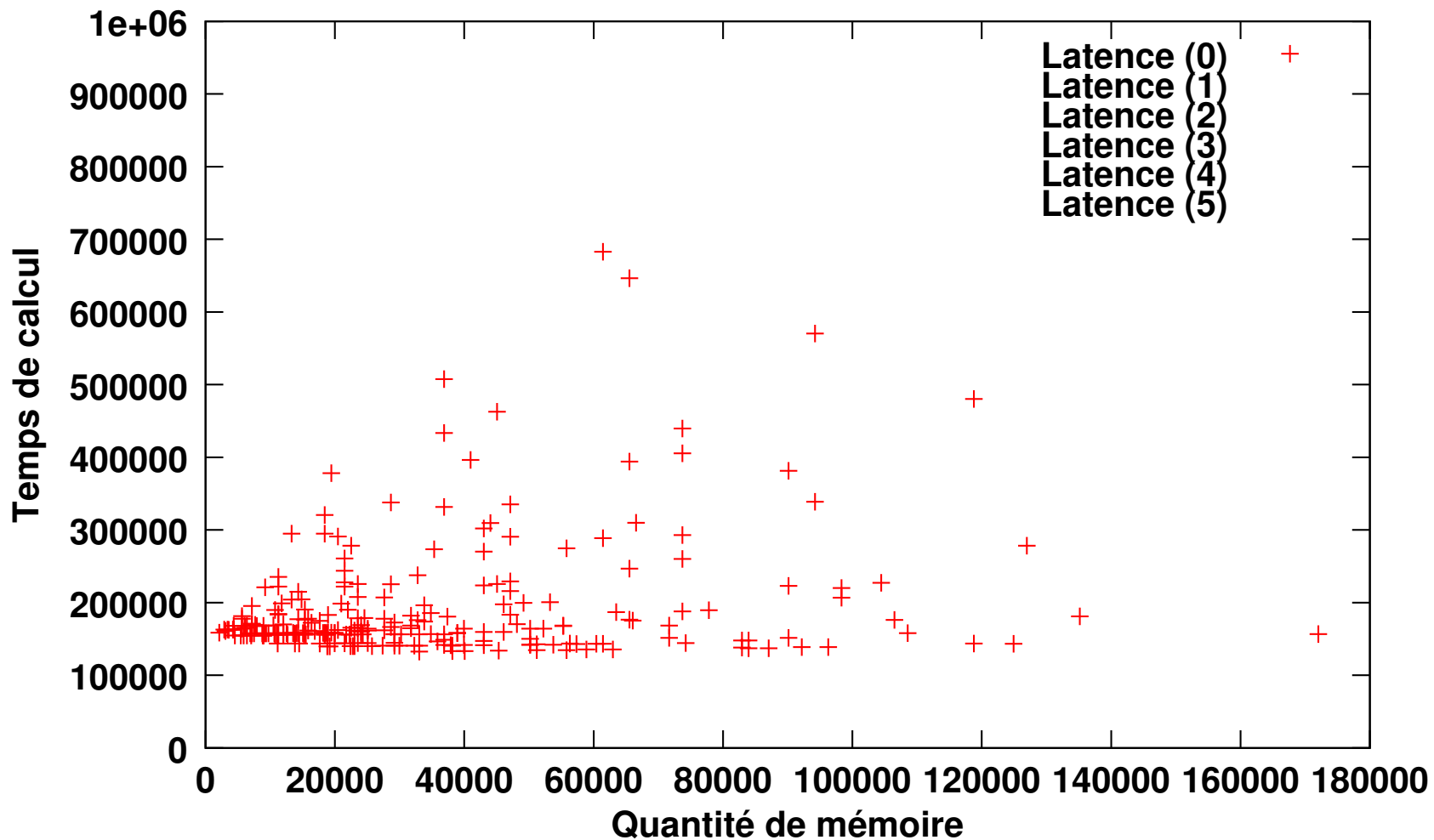
Résultats

Kernel	Input data type	Input data dimension	MMopt versus				
			CatapultC	Linear scheduling			
			Memory gain	Memory gain	Processed tile penalty	Traffic gain	
fisheye	image	2D	94	16	1.19	1.29	
fisheye	mipmap anisotropic	4D	>> 100	28	1.19	1.34	
polar	image	2D		34	1.26	1.27	
polar	mipmap anisotropic	4D		41	1.15	1.18	
polar	mipmap anisotropic	2D (flat)		568	1.08	1.21	
polar	mipmap isotropic	3D		21	1.02	1.38	
polar	mipmap isotropic	2D (flat)		1264	1.02	1.37	
pseudolog	image	2D		24	1.20	1.19	
pseudolog	mipmap anisotropic	4D		63	1.20	1.20	
pseudolog	mipmap anisotropic	2D (flat)		344	1.09	1.37	
Average					32	1.14	1.28

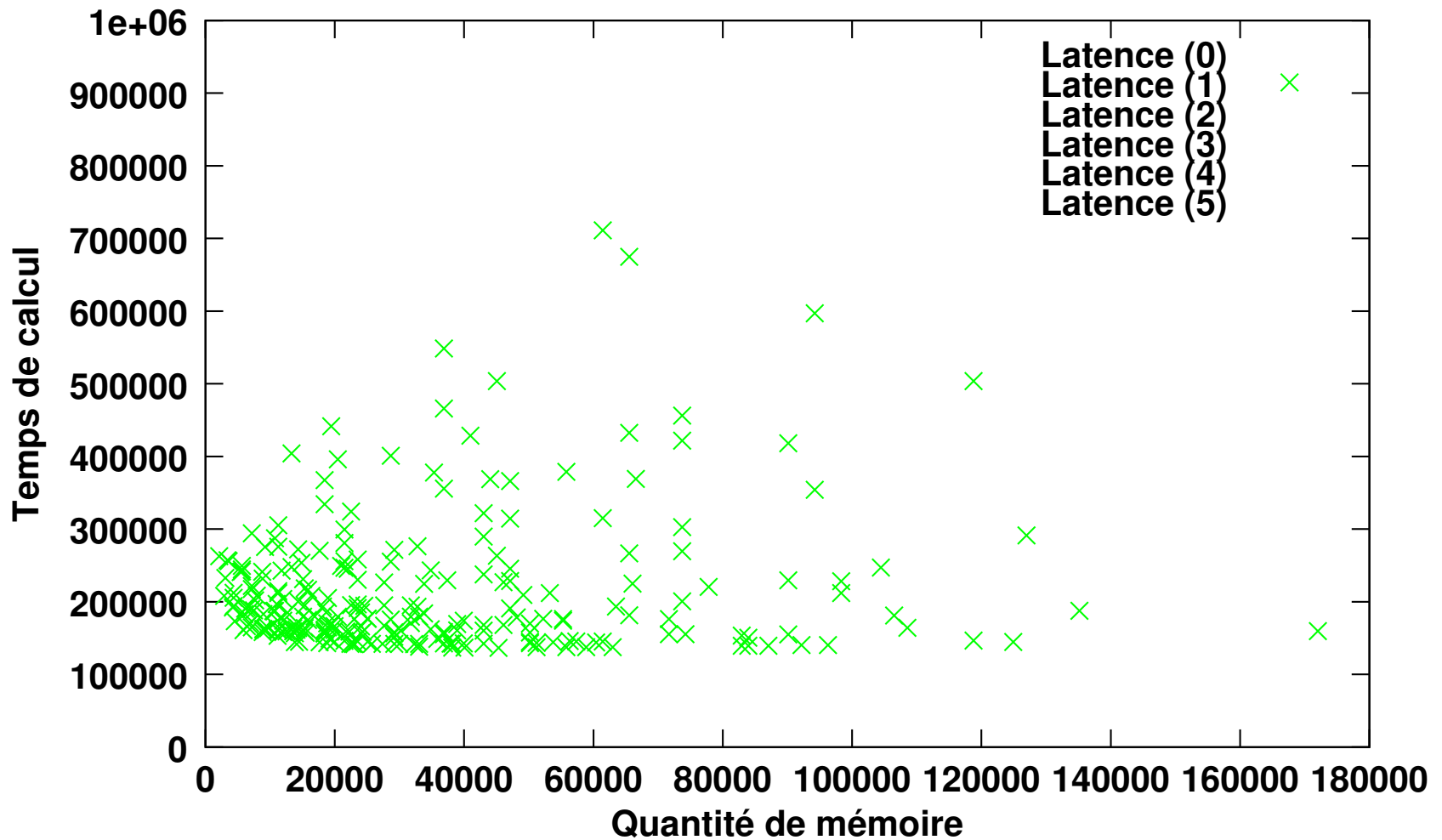
Détail du processus d'optimisation (trans. polaire)



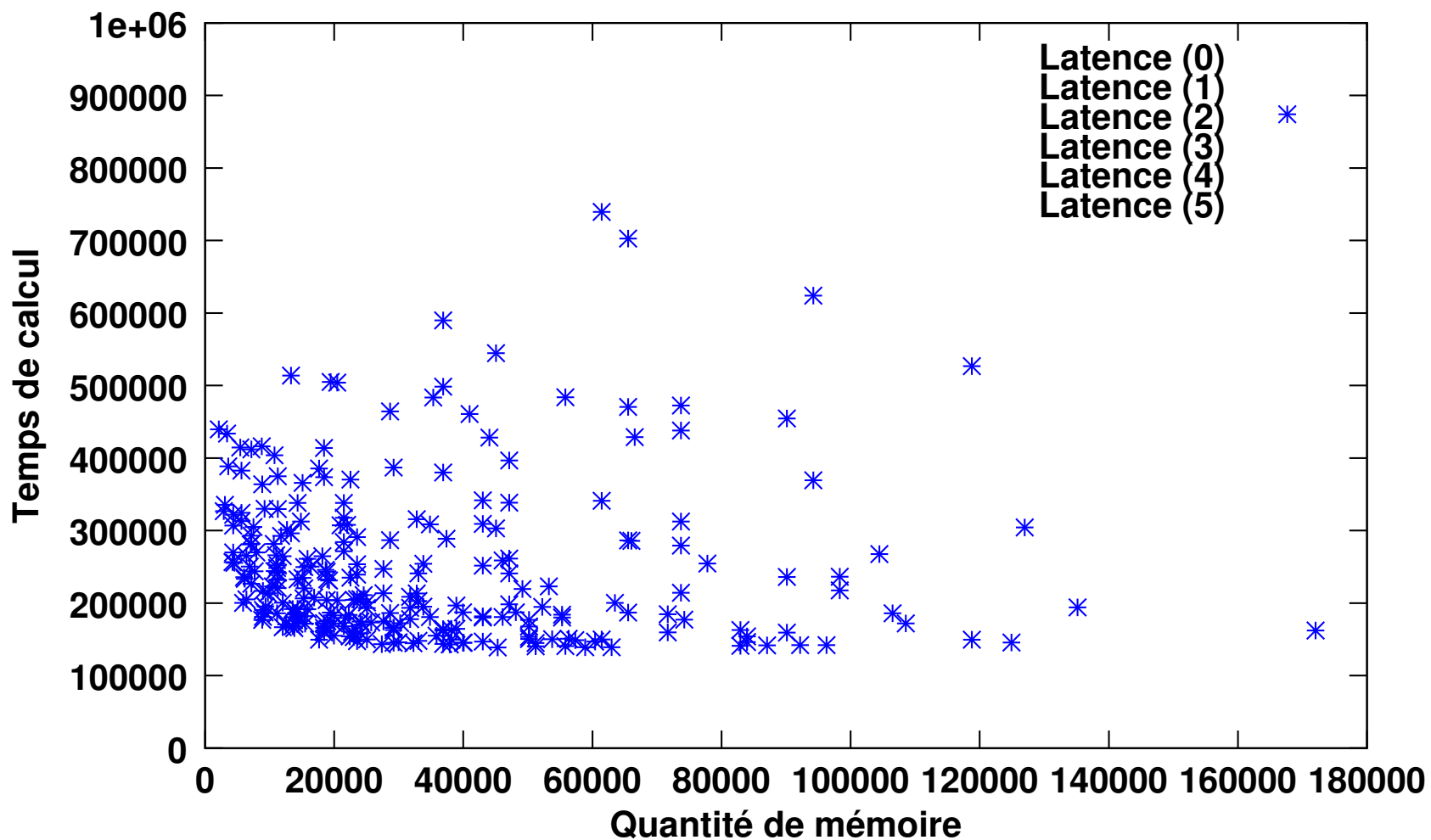
Choisir un tuilage ?



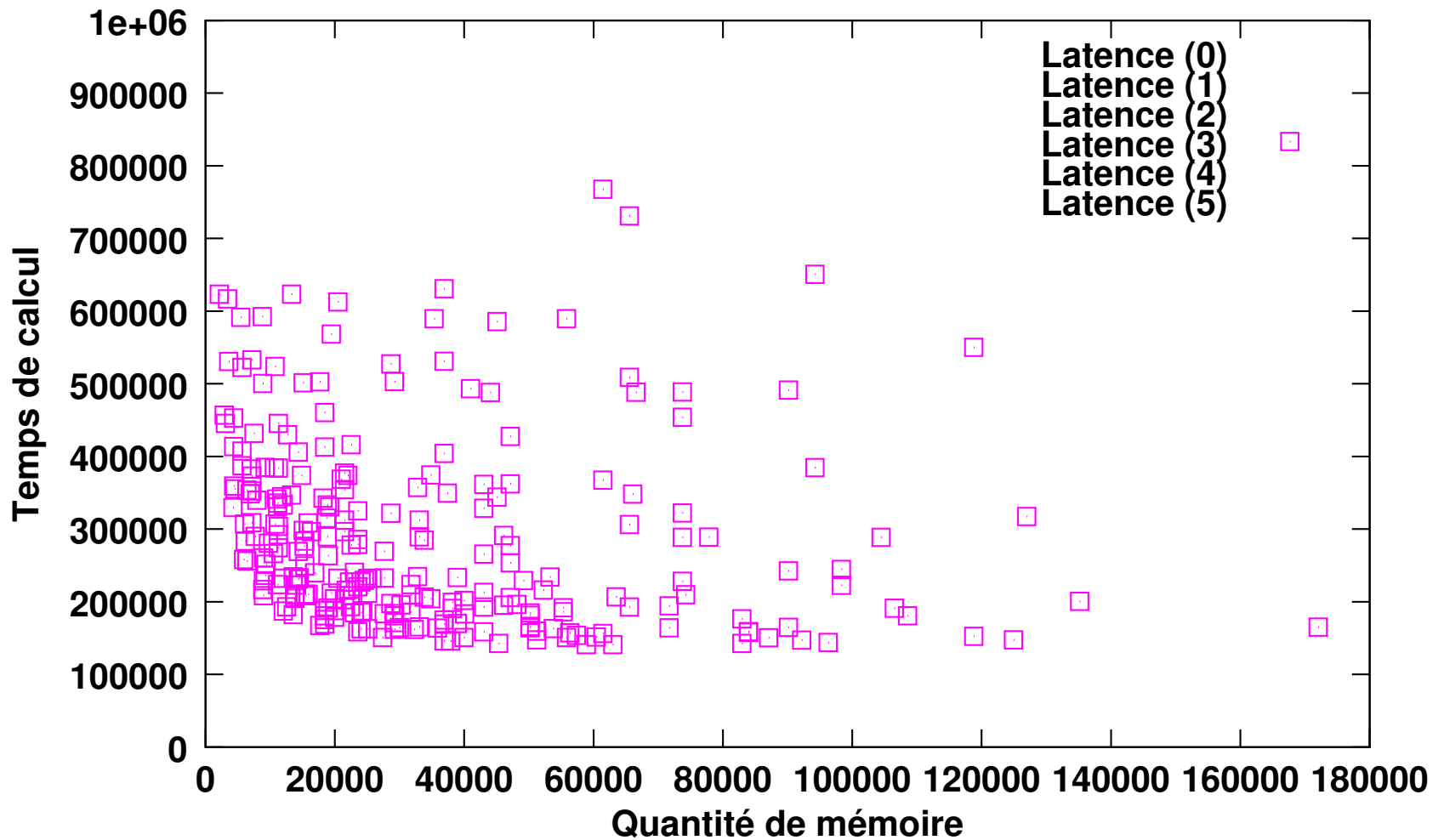
Choisir un tuilage ?



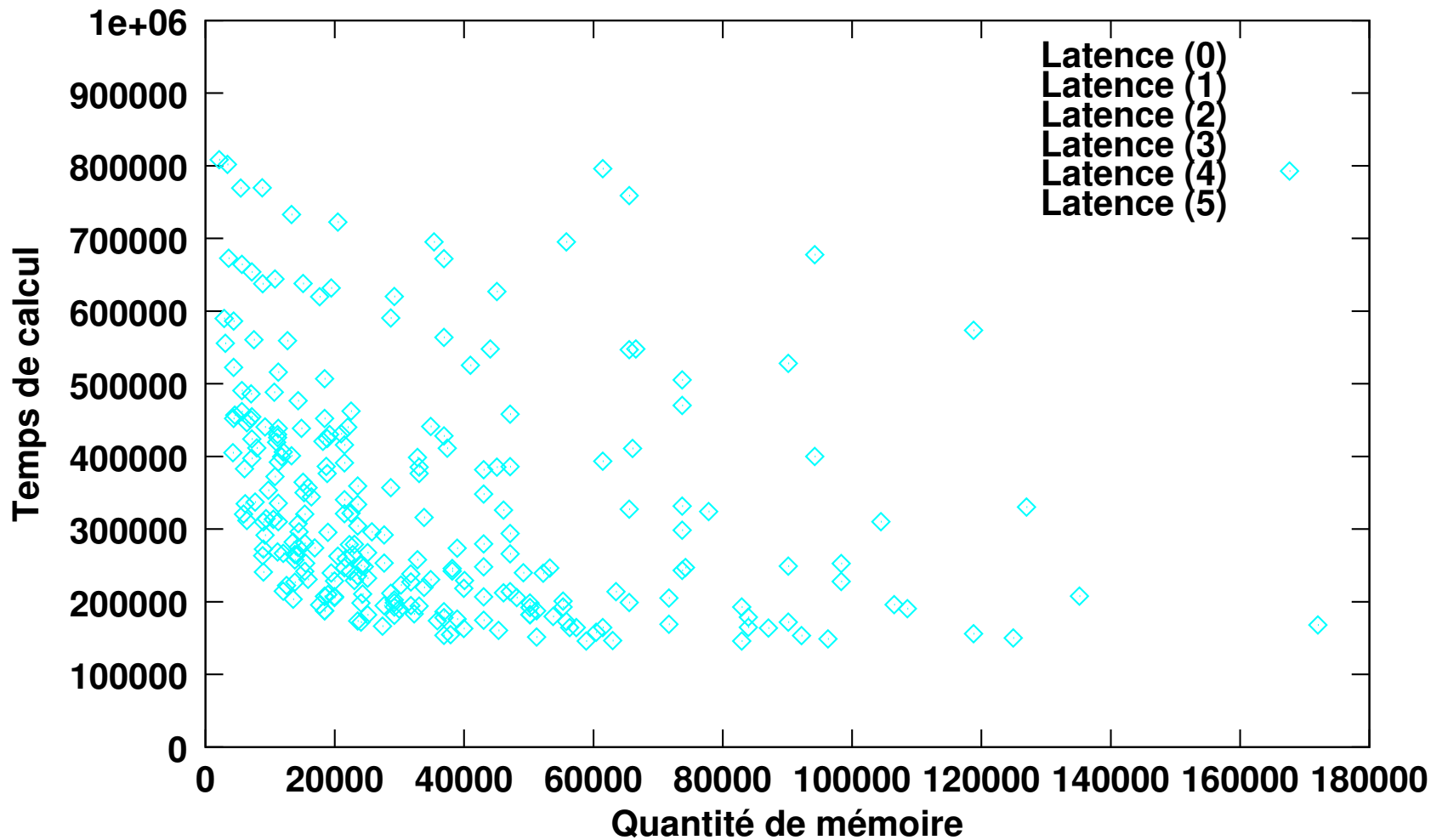
Choisir un tuilage ?



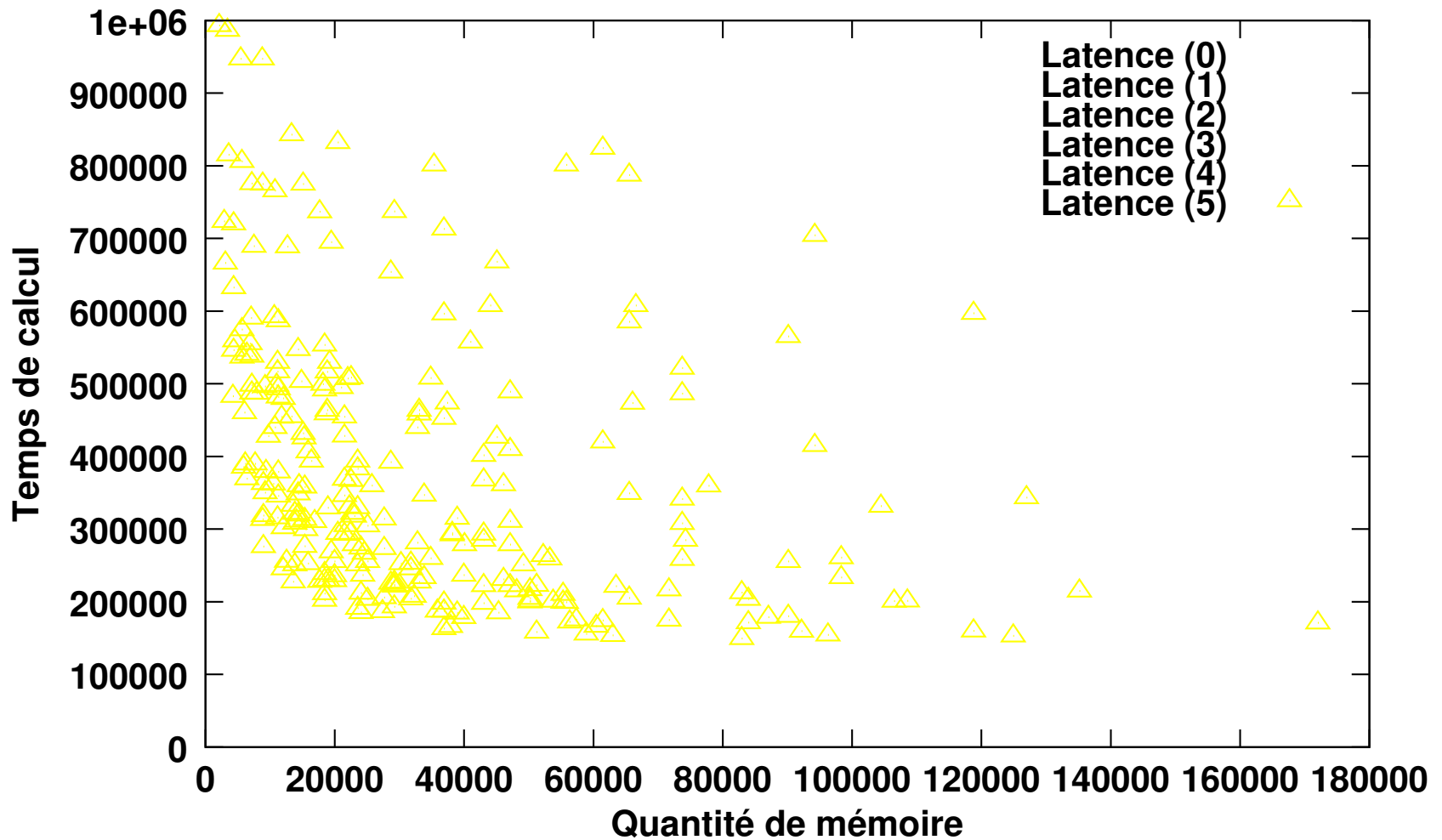
Choisir un tuilage ?



Choisir un tuilage ?



Choisir un tuilage ?

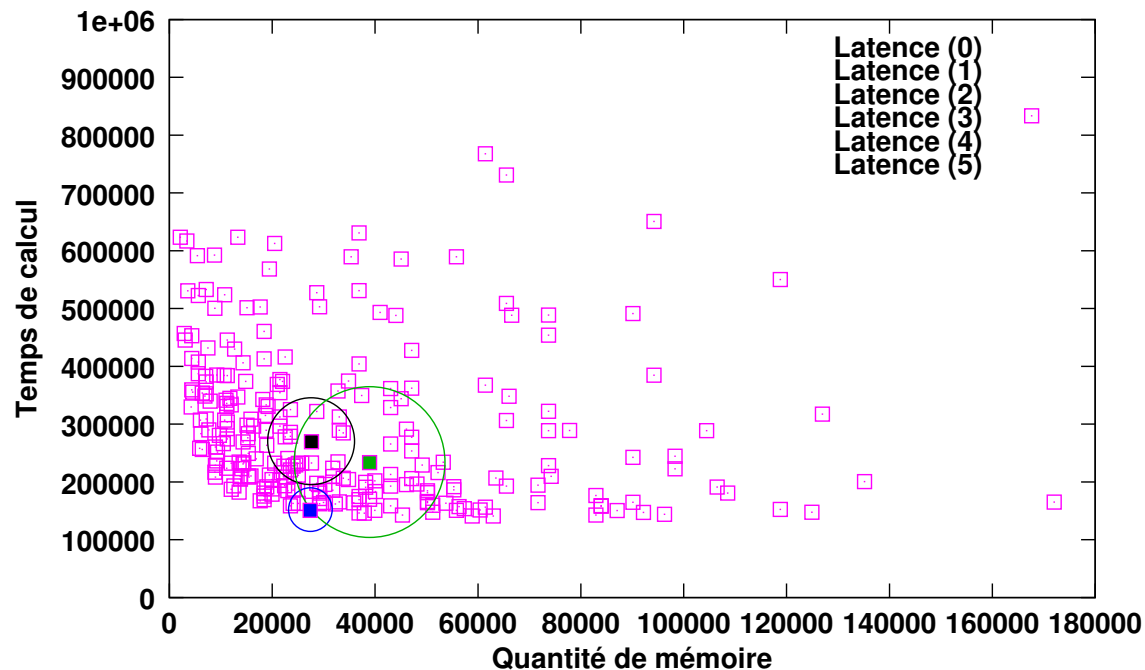


Effet de l'optimisation sur les courbes pareto ?

Les courbes pareto “initiales” sont obtenues avec un ordonnancement linéaire, sans optimisation.

Le processus d'optimisation modifie le temps de calcul et la quantité de mémoire.

- ? Comment trouver des courbes pareto “optimales” le plus rapidement possible ?
- ?



Systemes multi-kernel

Optimisation des systemes constitues de plusieurs unites de traitement :

- ☆ Comment les représenter ?
- ☆ Comment optimiser les ressources de “communication” (buffers intermédiaires) ?

Conclusion

- ☆ Le processus d'optimisation permet des compromis temps/surface “meilleurs” que les méthodes classiques.
- ☆ Intérêt des transformations source-to-source pour la HLS et hiérarchie mémoire :
 - Les outils HLS n'ont pas à gérer les aspects systèmes si on peut les intégrer dans le code source.

Journées SEMBA 2011

HLS & Hiérarchie mémoire

S. Mancini

Plan Détaillé

- ☆ Contexte
- ☆ Traitement de lois d'accès non-linéaires
- ☆ Problématique
- ☆ Vue générale du flot
- ☆ Architecture cible
- ☆ Séquencement
- ☆ Optimisations
- ☆ Analyse
- ☆ Gestion de la disparité
- ☆ Compromis disponibles
- ☆ Benchmarks
- ☆ Mipmap ?
- ☆ Résultats
- ☆ Détail du processus d'optimisation (trans. polaire)
- ☆ Choisir un tuilage ?
- ☆ Effet de l'optimisation sur les courbes pareto ?
- ☆ Systèmes multi-kernel
- ☆ Conclusion