Journée SOC-SIP 15-11-2012

Gestion des données pour les noyaux non-linéaires dans un flot de conception HLS

S. Mancini & F. Rousseau

iima
CNRS - Grenoble INP - UJF

Grenoble INP

Contexte : la vision embarquée

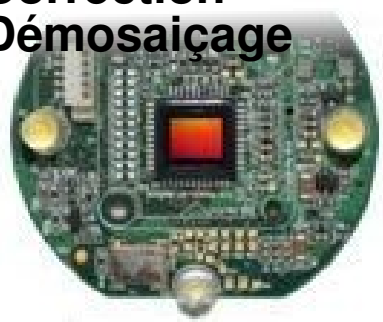
Optique

Acquisition & Prétraitements

Correction Démosaïçage

Supervision

Détection Reconnaissance

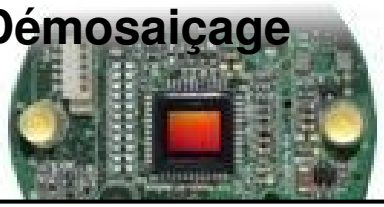


Contexte : la vision embarquée

Optique

Acquisition & Prétraitements

Correction Démosaïçage



Supervision

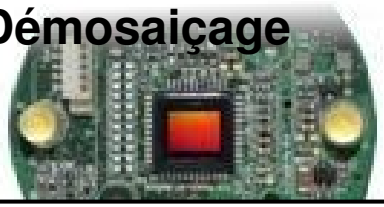
Détection Reconnaissance

Contexte : la vision embarquée

Optique

Acquisition & Prétraitements

Correction
Démosaïçage



Supervision

Détection
Reconnaissance

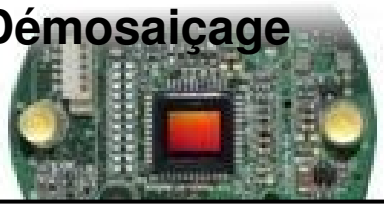
Haute performance
Basse consommation

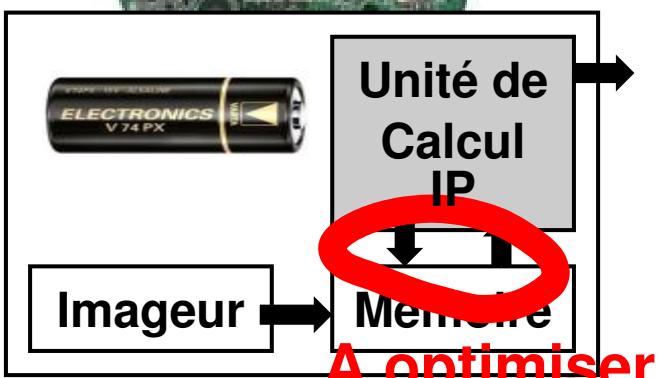
Contexte : la vision embarquée

Optique

Acquisition & Prétraitements

Correction
Démosaïçage





Supervision

Détection
Reconnaissance

Haute performance
Basse consommation

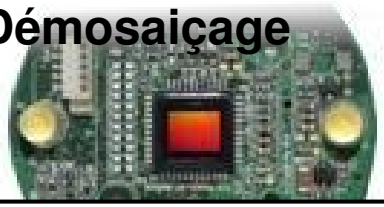
Contexte : la vision embarquée

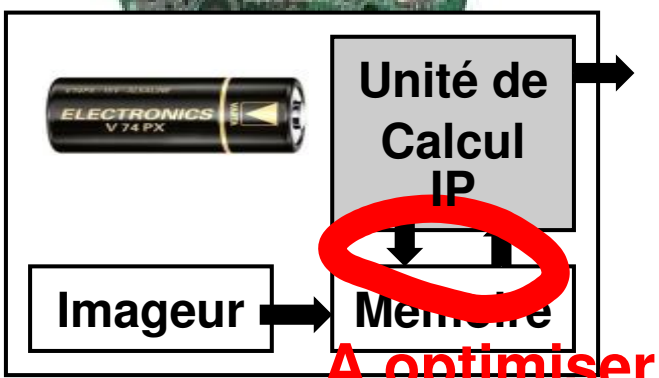
Optique

Acquisition & Prétraitements

Supervision

Correction Démosaïçage





Détection Reconnaissance

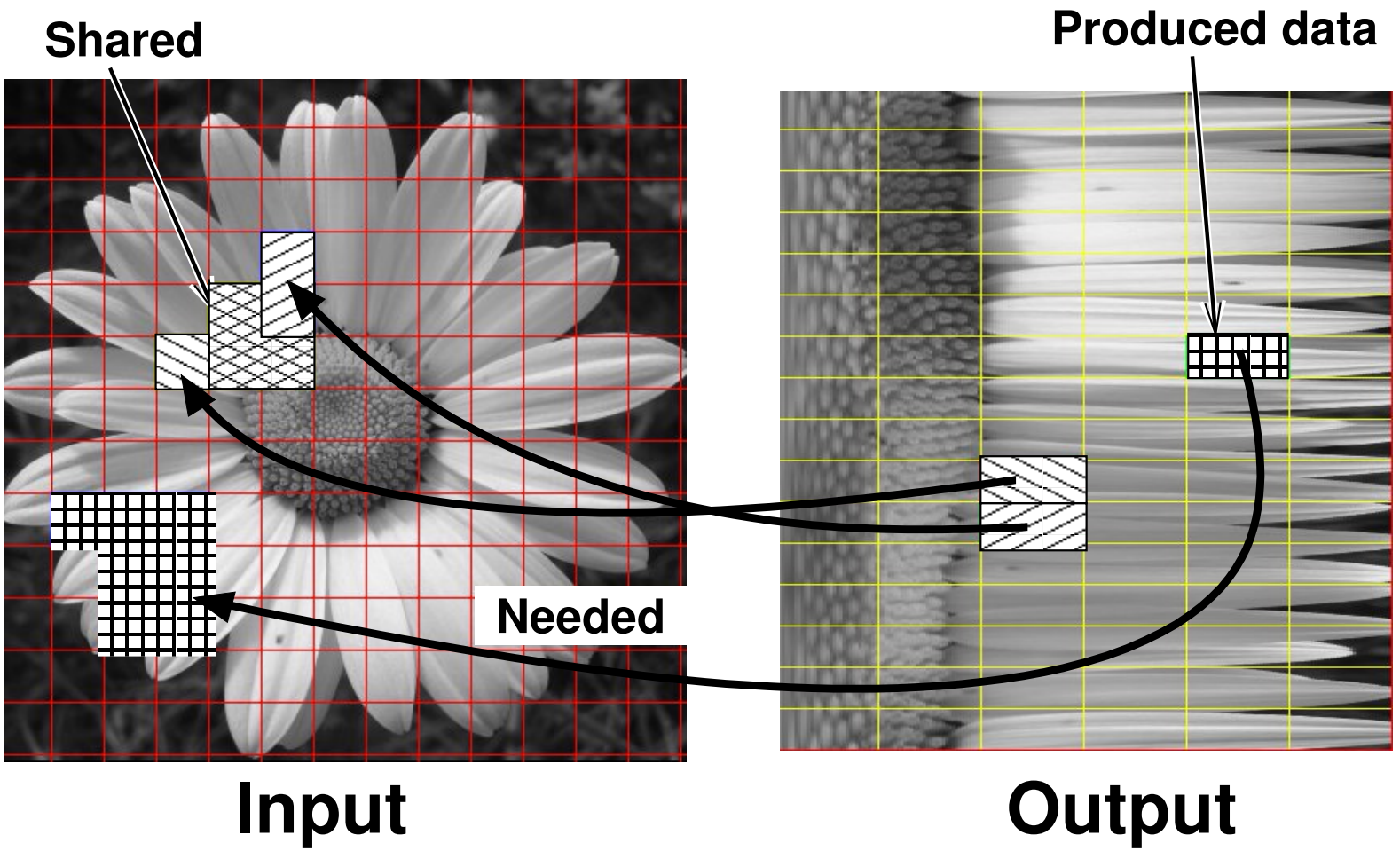
Haute performance
Basse consommation

Enjeux : concevoir des systèmes efficaces et à faible coût

Traitement de lois d'accès non-linéaires

Applications:

- Système visuel "humain", ego-motion, fisheye, auto-focus, etc . . .



Méthodes de conception classiques

☆ **Conception "RTL"** (Register Transfer Level)

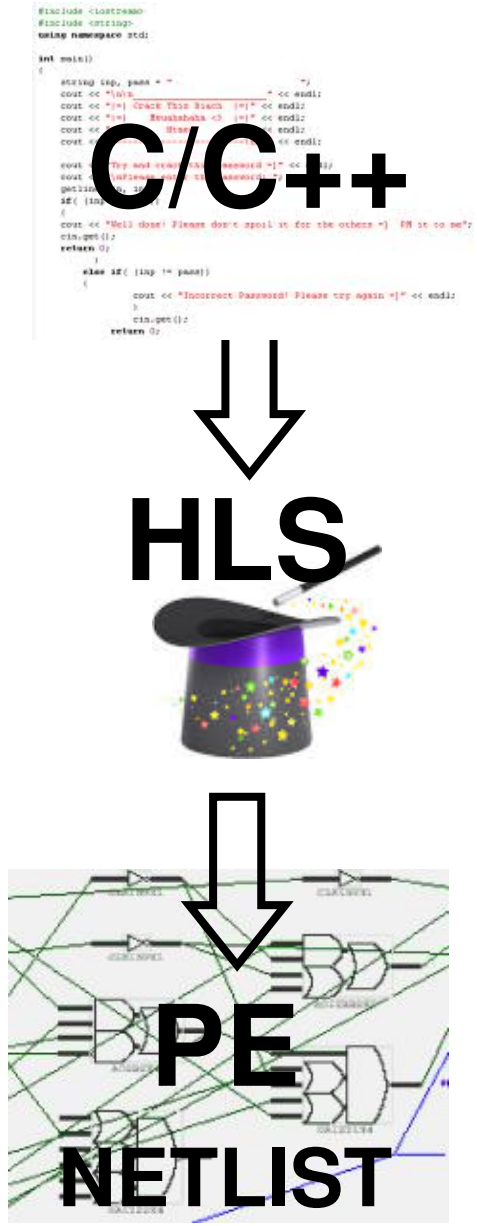
Le circuit est conçu "à la main", à l'aide de langages adaptés.

↗ Précis ↘ Long, exploration d'architecture & gestion de la complexité difficiles

☆ **Conception "HLS"** (High Level Synthesis)

Le concepteur décrit le traitement par un programme C/C++. La HLS le "transforme" en circuit (**PE** - Processing Element).

↗ Conception très rapide



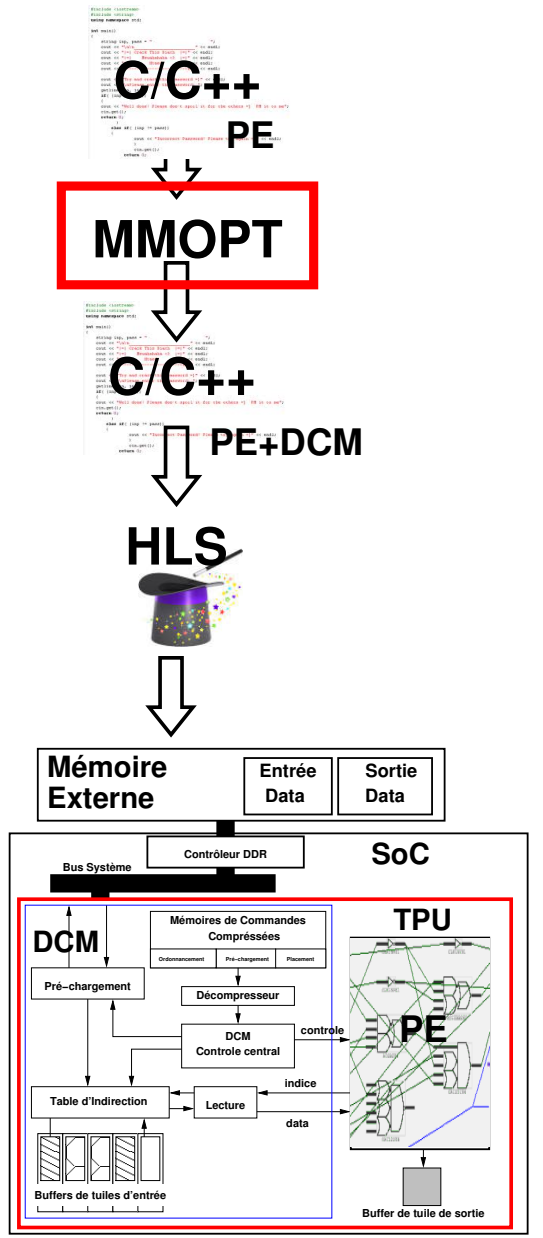
Problèmes:

↘ Les accès complexes **non-linéaires** aux données en mémoire ne sont pas gérés

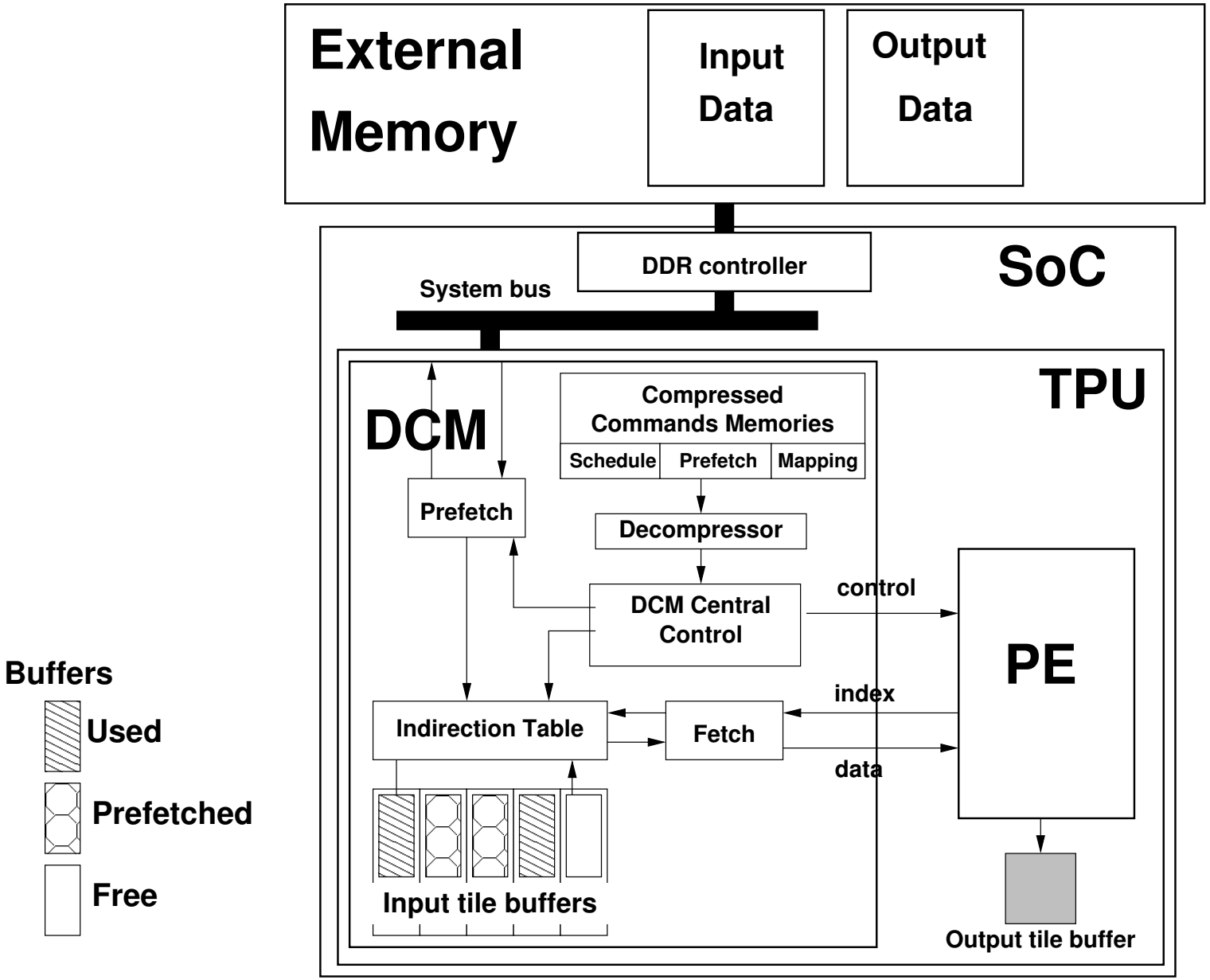
↘ Les caches standard ne sont pas adaptés

Solution proposée

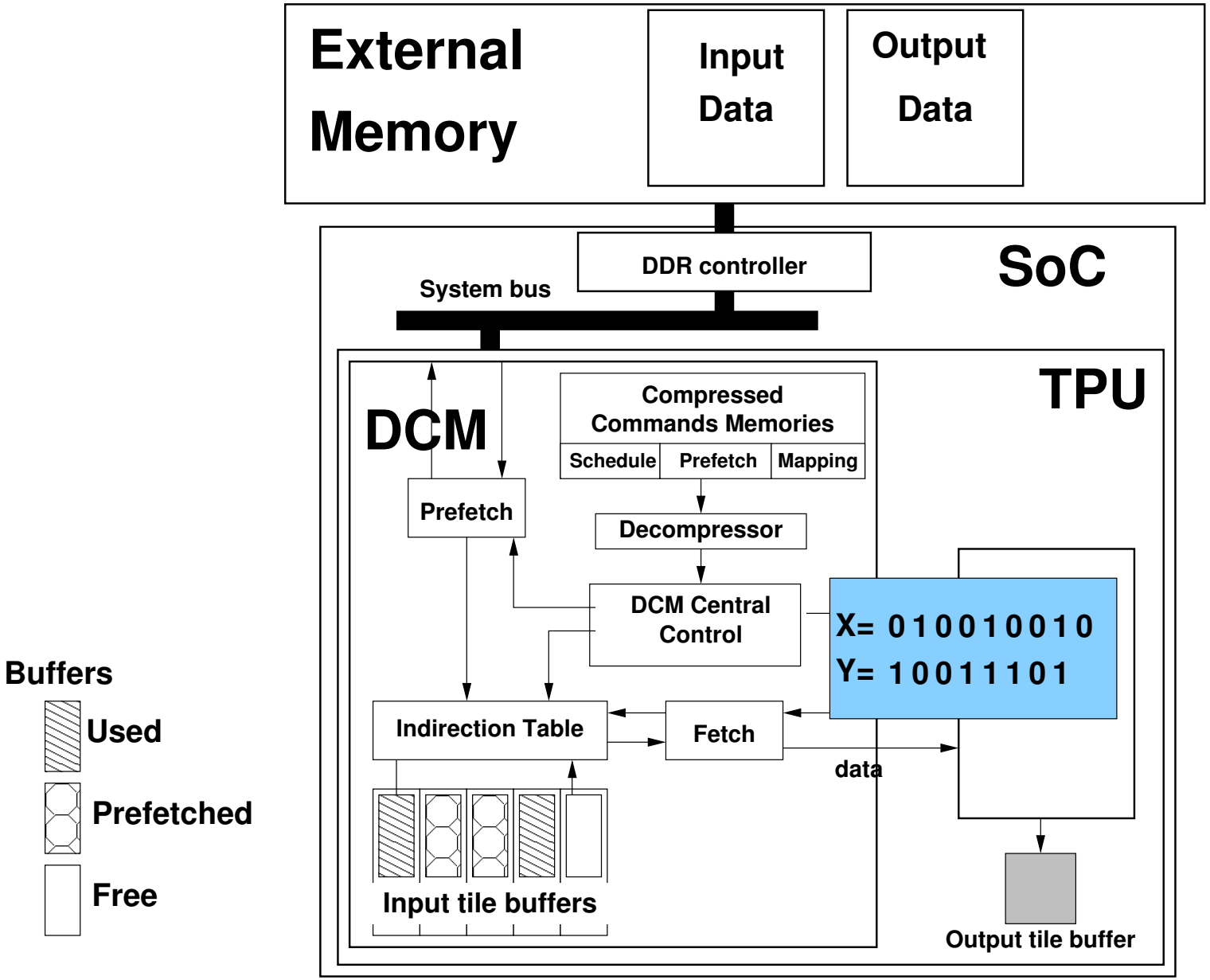
- **Principe**: **Optimiser la gestion des données** par incorporation **automatique** d'un contrôleur mémoire dans un flot de conception par synthèse de haut niveau (**HLS**).
- **Solution**: Paramétrer un contrôleur générique des données et du séquencement des calculs (**DCM**) à partir d'une description pour la HLS du traitement (**PE**).
- **Mise en oeuvre**: l'**atelier** MMopt
- **Etat des travaux**:
  - Gains : réduction mémoire interne 30 fois (coût), réduction traffic mémoire 30% (énergie)
  - Principe de base validé, outil prototype
  - Tests sur plusieurs circuits de tailles "raisonnables"



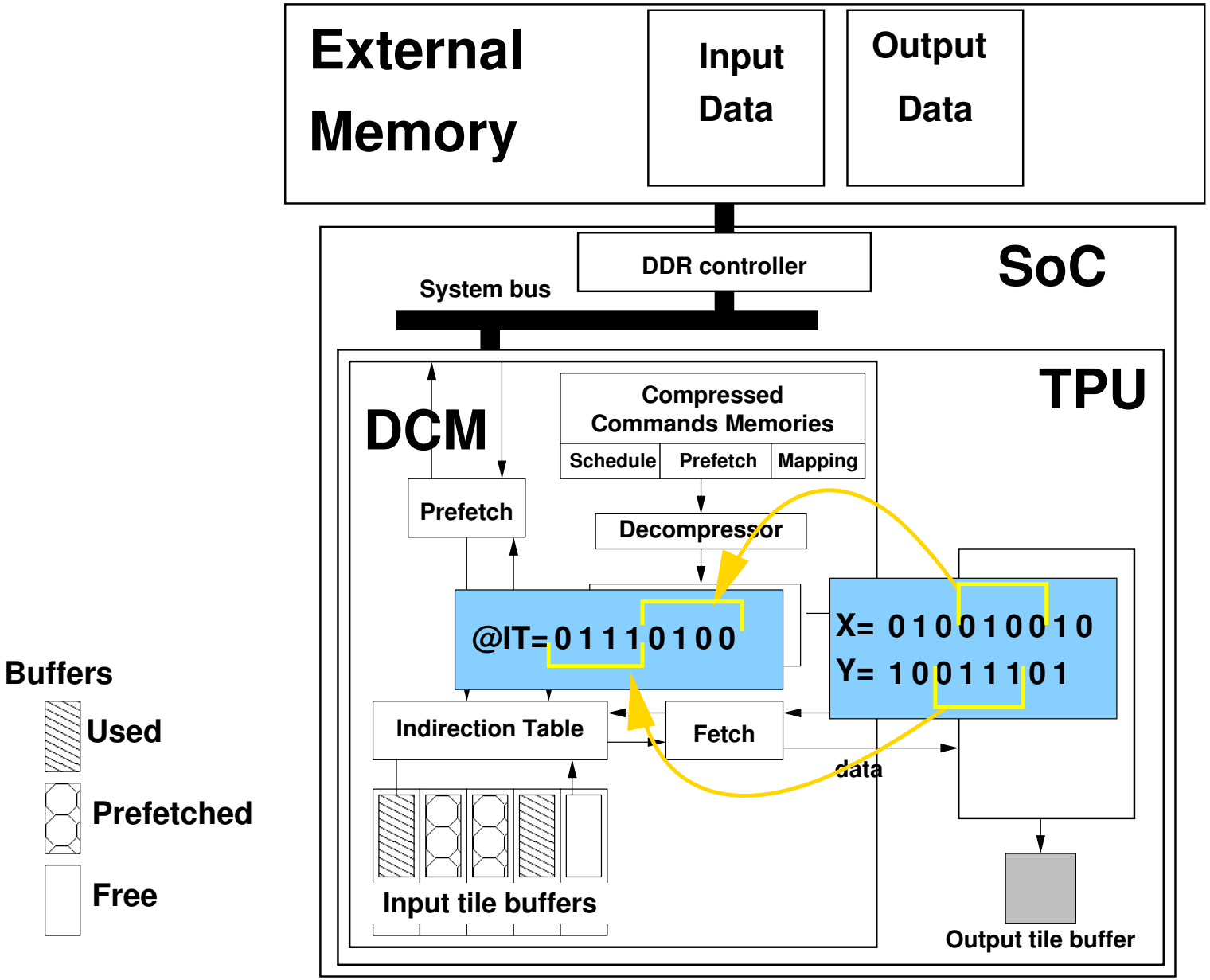
Architecture générique



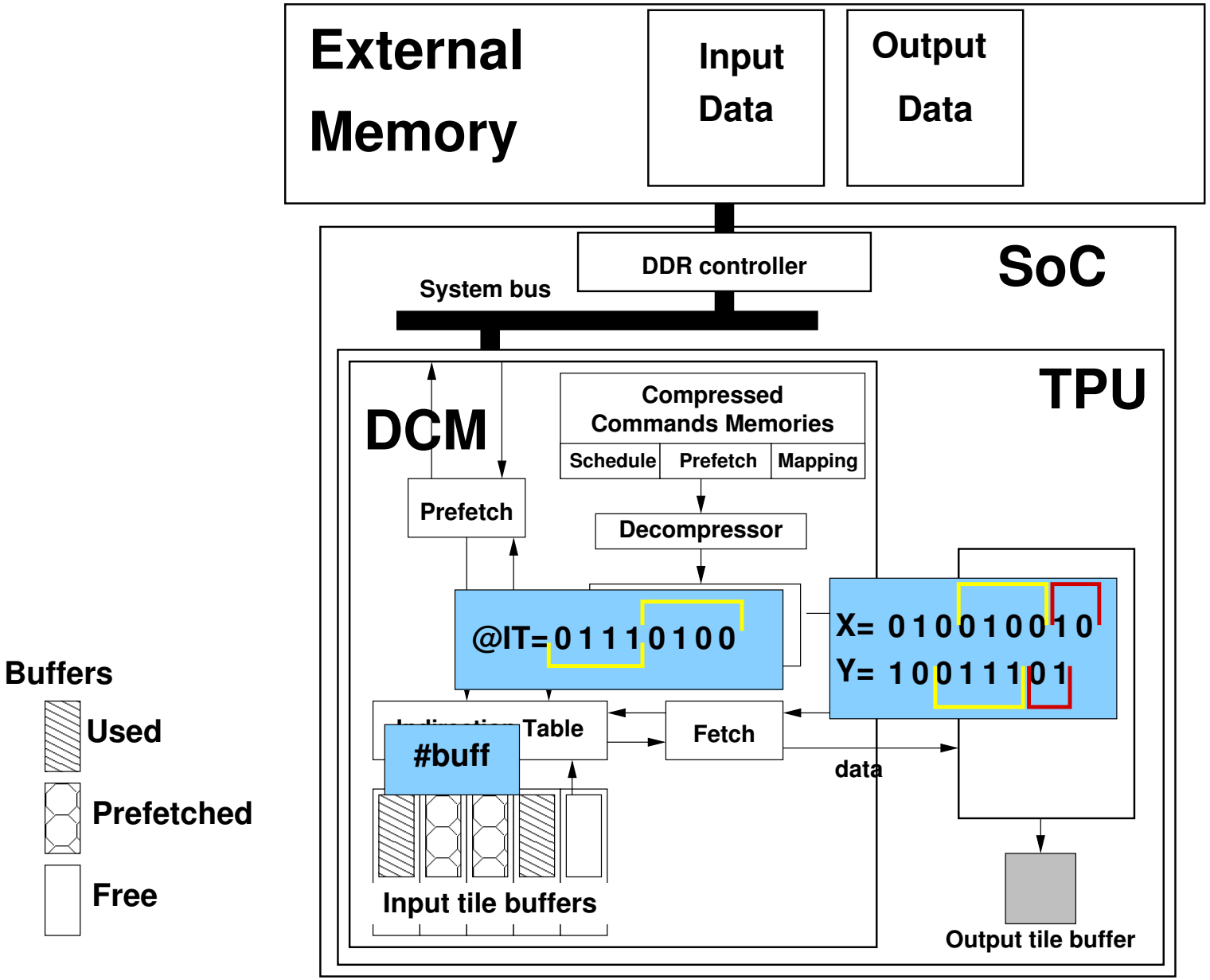
Architecture générique



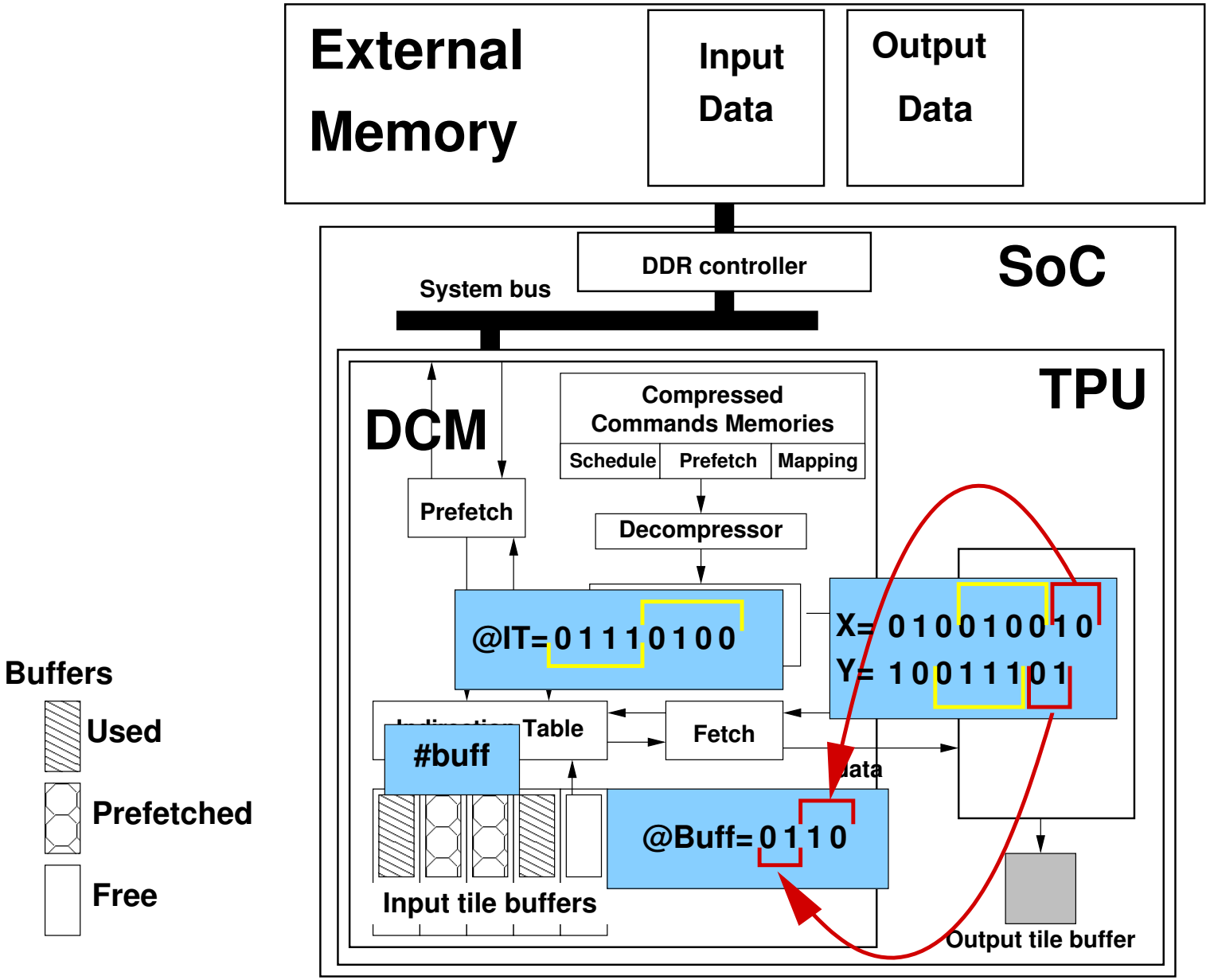
Architecture générique



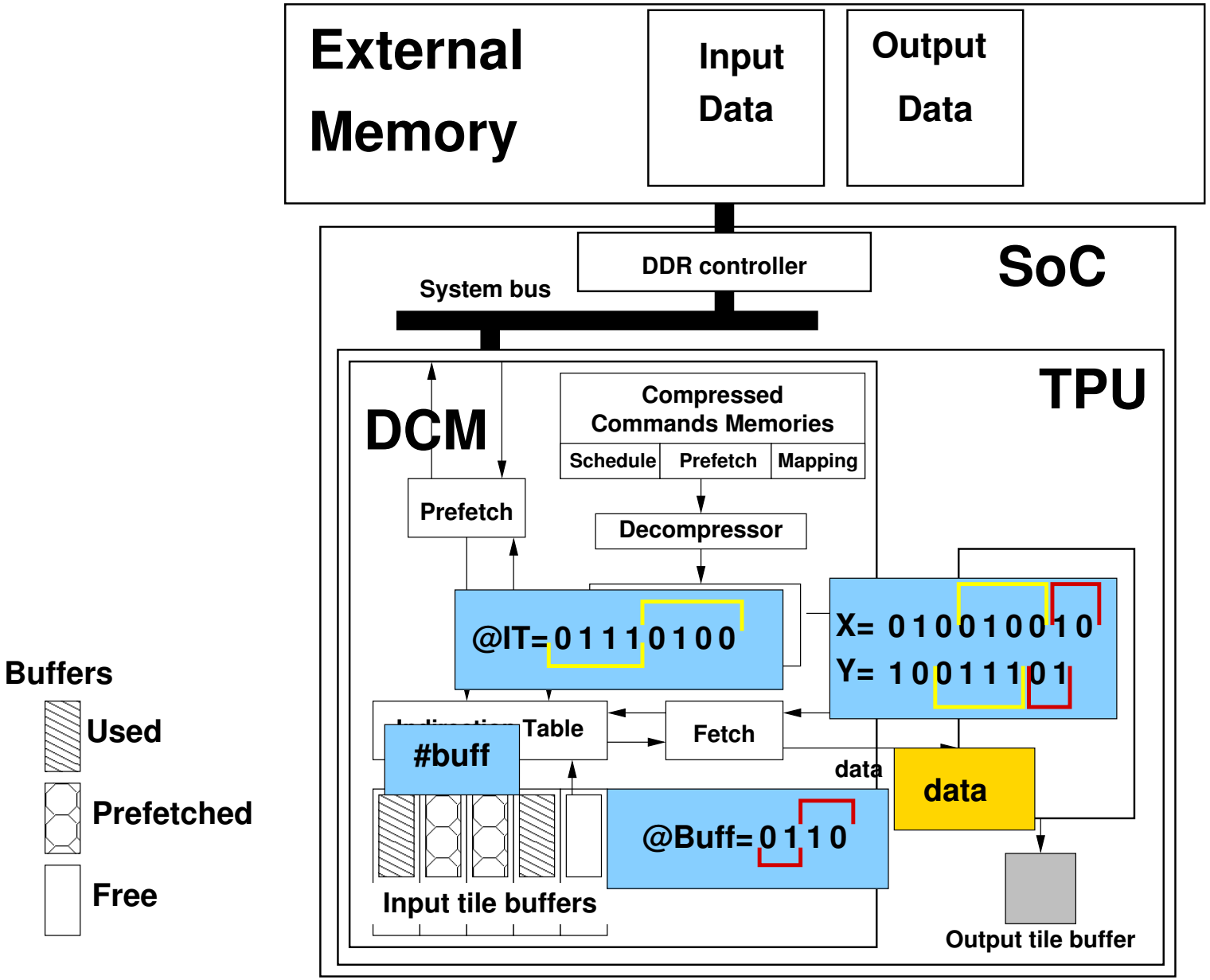
Architecture générique



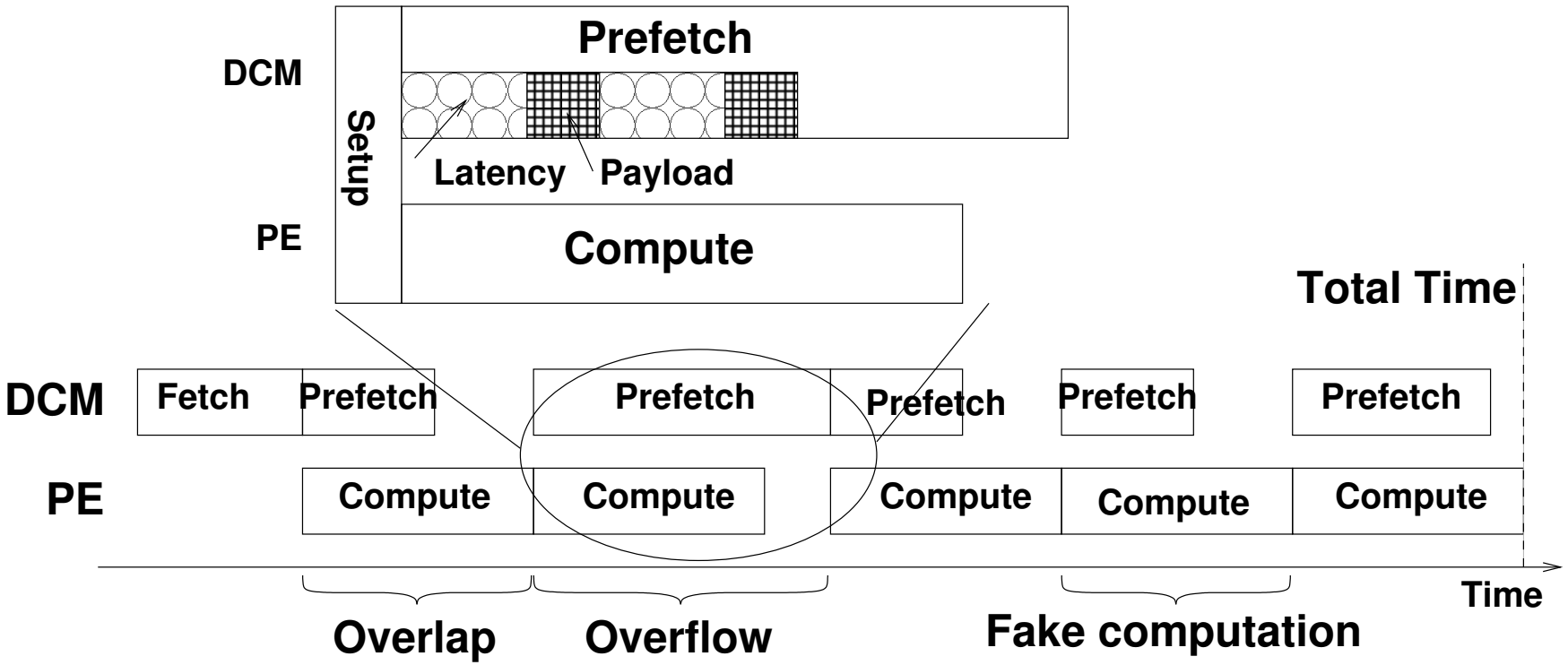
Architecture générique



Architecture générique



Séquencement



Optimisation de la gestion des données

- Critères à optimiser
  - Traffic depuis la mémoire centrale
  - Temps de calcul total
  - Quantité de mémoire interne
    - Nombre de buffer
    - Taille de la table IT
    - Taille des mémoires de commandes
- Leviers
  - Taille des tuiles
  - Ordonnancement du calcul des tuiles de sortie
  - Ordonnancement du pré-chargement
  - Compression des tables (LRE)

Minimisation du débit

Le traffic à la mémoire centrale est minimisé en optimisant l'ordonnancement des tuiles de sortie.

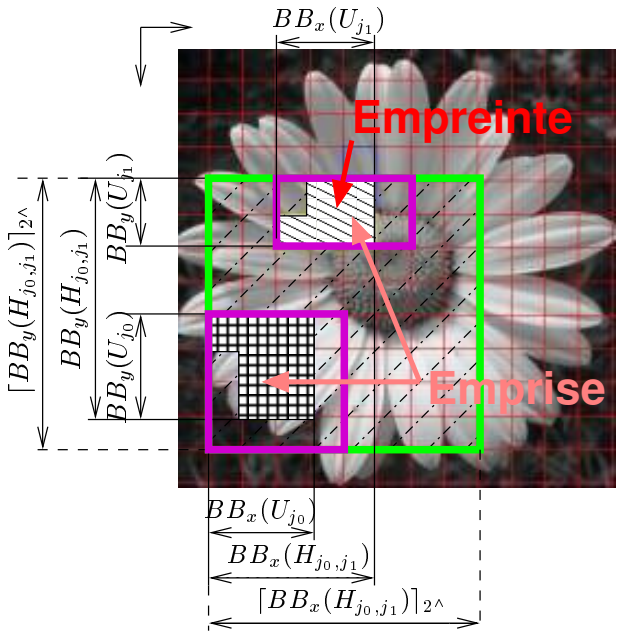
Cette optimisation équivaut à résoudre un TSP asymétrique, sur un graphe orienté dont les arêtes valent le coût de chargement de tuile en tuile (asymétrique).

- ☞ Peut être résolu avec des heuristiques connues.

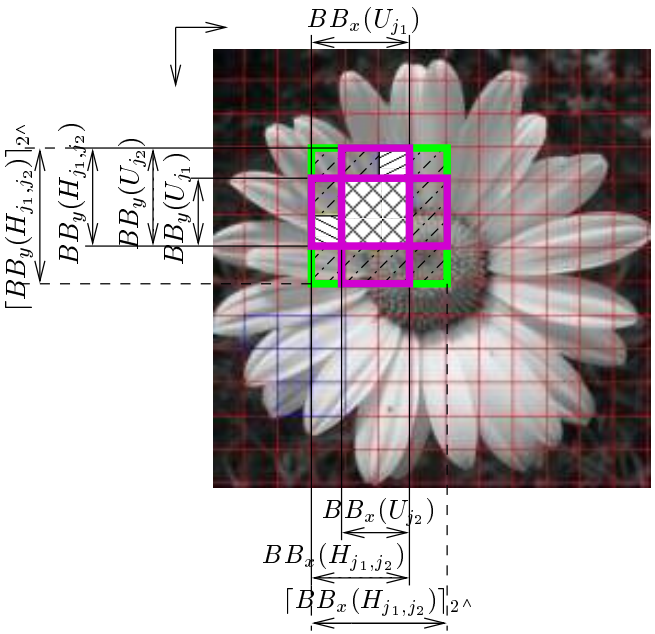
440 tuiles chargées

390 tuiles chargées

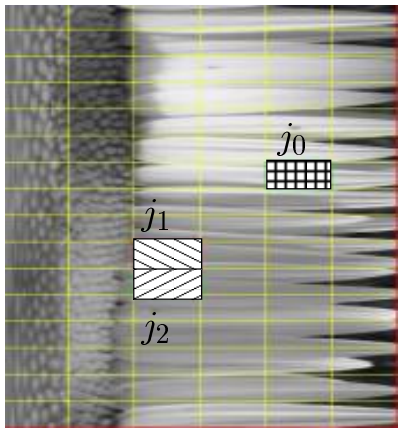
Calcul des ressources mémoire



a) Calcul j_0 (pré-charge pour j_1)



b) Calcul j_1 (pré-charge pour j_2)

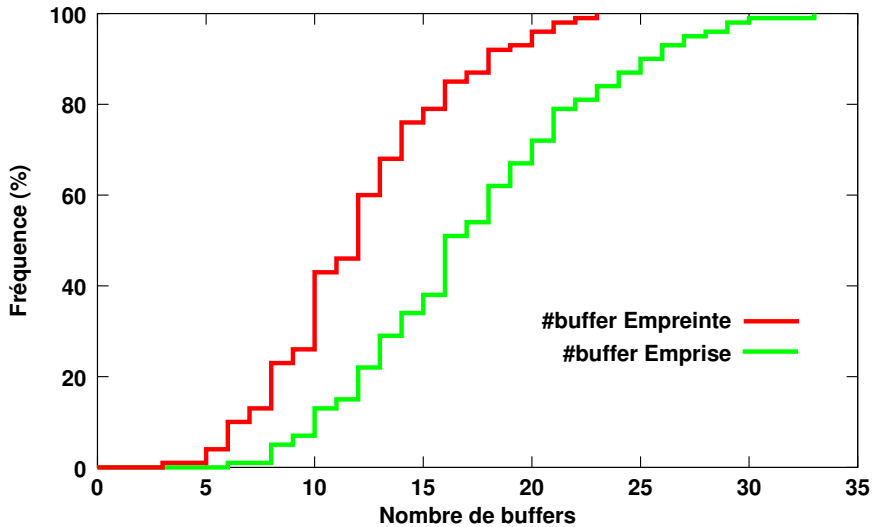


c) Sortie

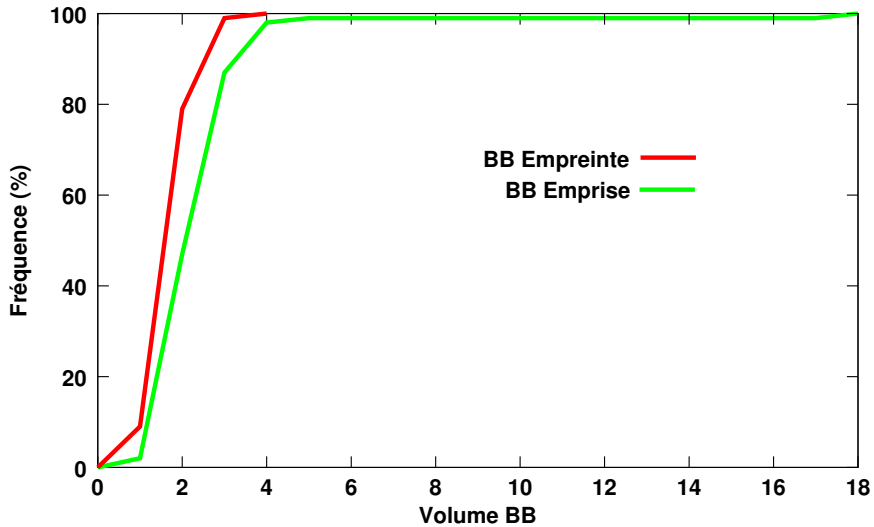
Séquencement $(j_0, j_1, j2)$ nécessite:

- ☆ 12 buffers
- ☆ table IT de 8x8=64 mots

Analyse



Histogramme du nombre de buffers



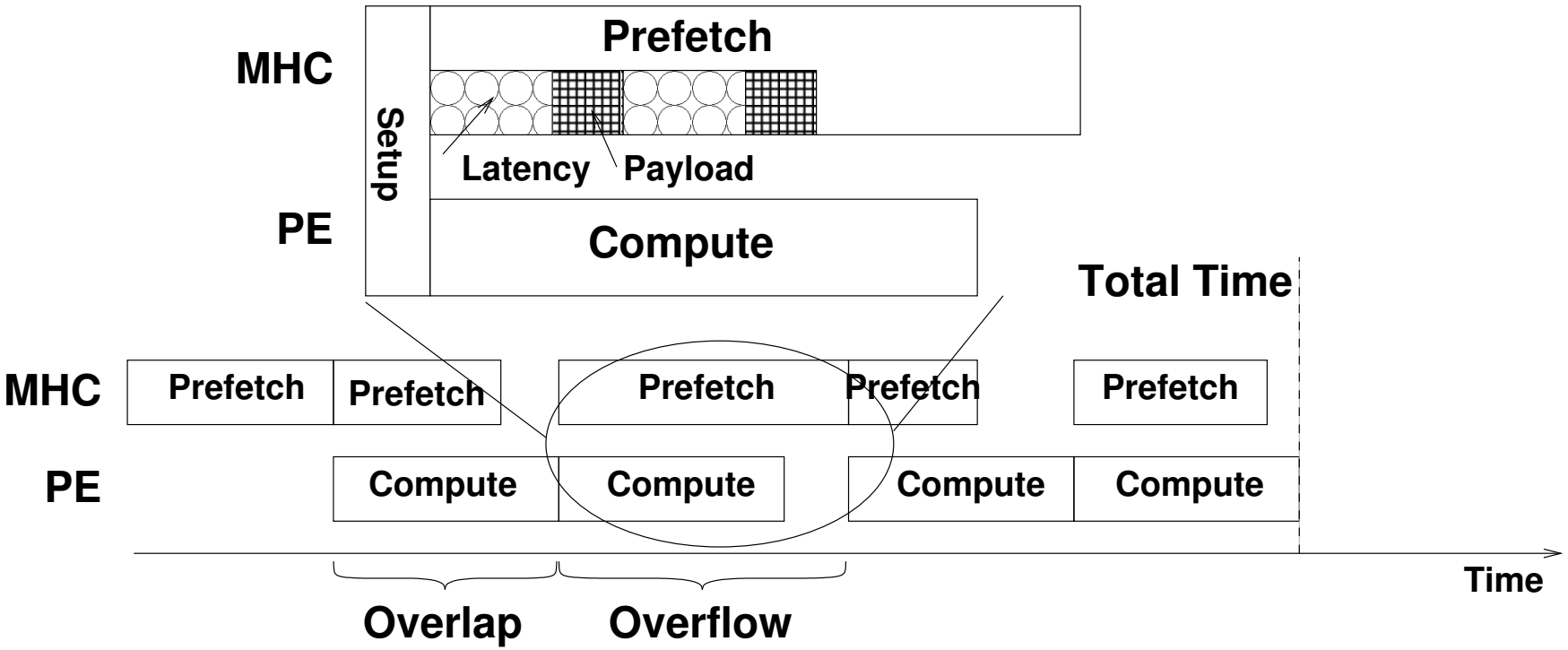
Histogramme de taille de boite englobante (BB)

Observation:

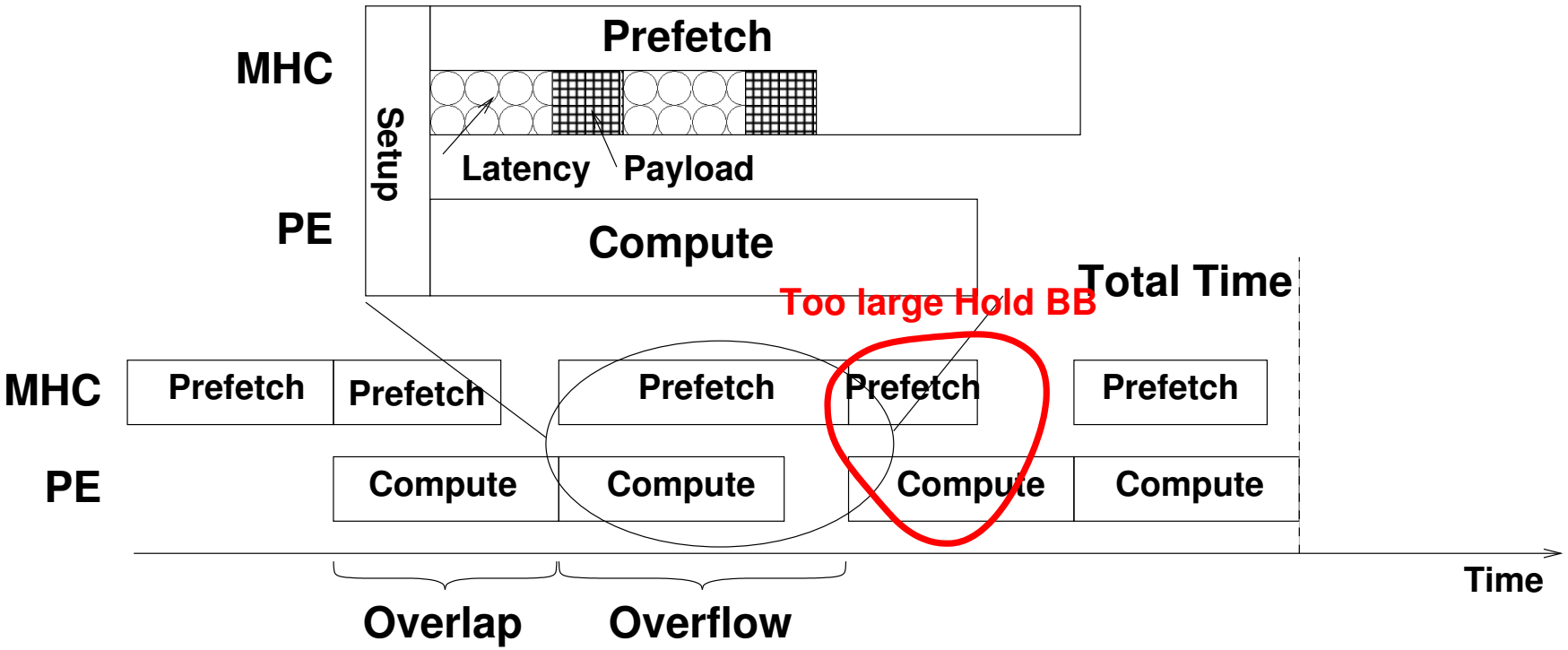
- Indépendamment de l'ordonnancement, les ressources minimum correspondent au pire cas de l'empreinte
- Après ordonnancement, les ressources maximum sont dimensionnées sur le pire cas de l'emprise

☞ Idée: fixer les ressources entre ces deux bornes et faire la précharge lorsque c'est possible

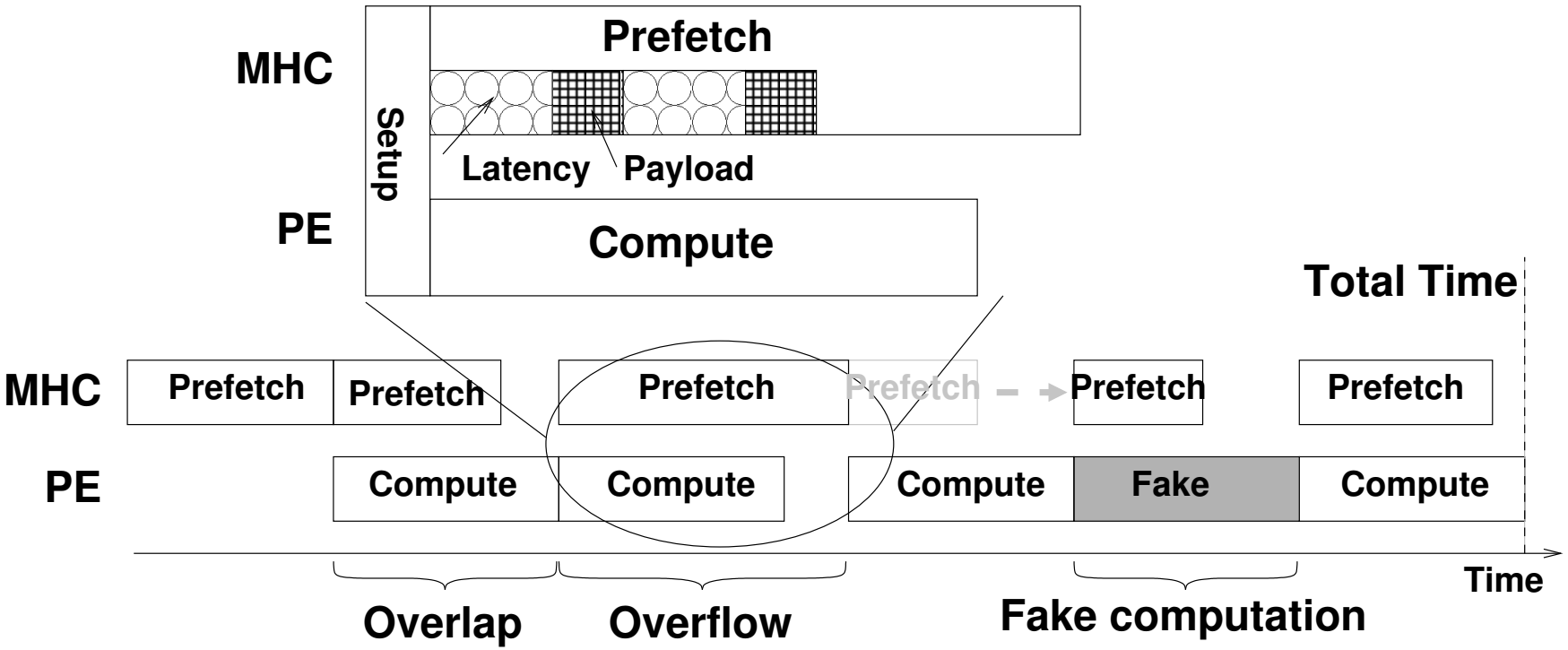
Gestion de la disparité



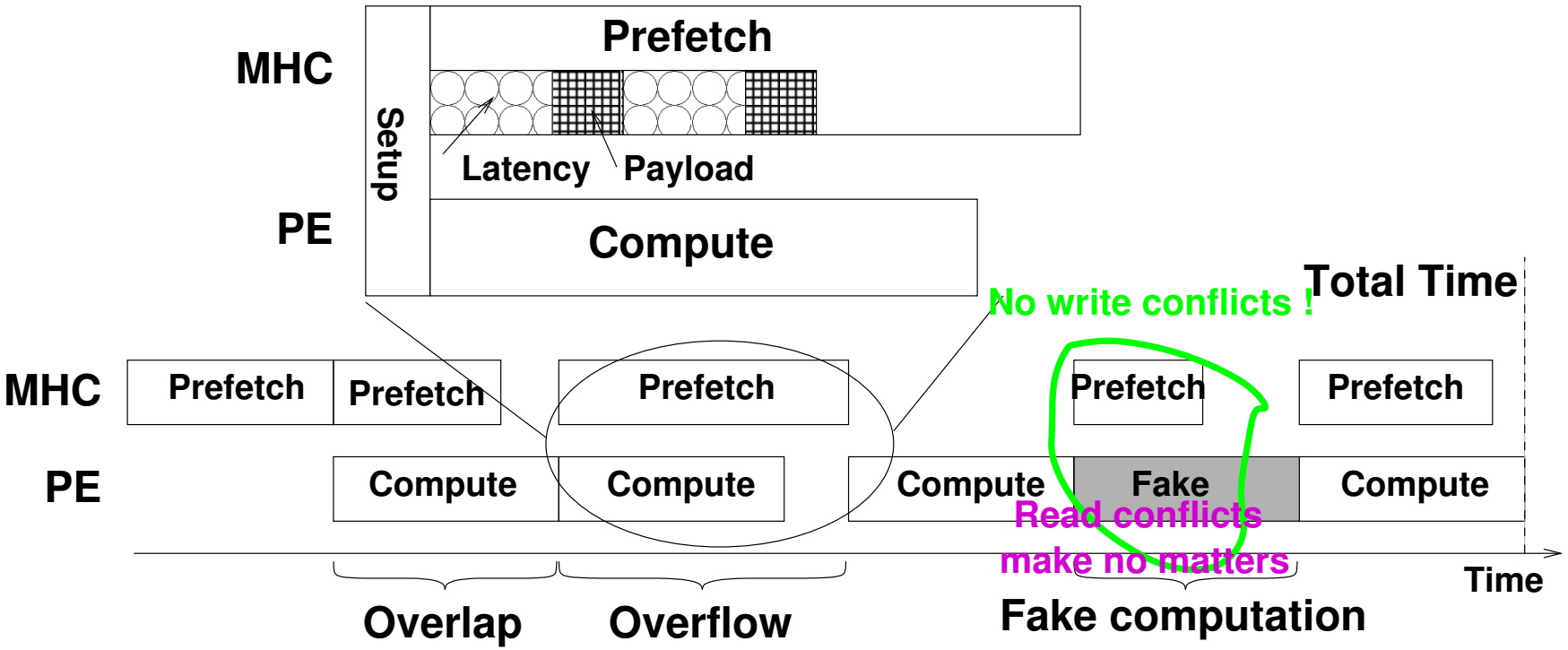
Gestion de la disparité



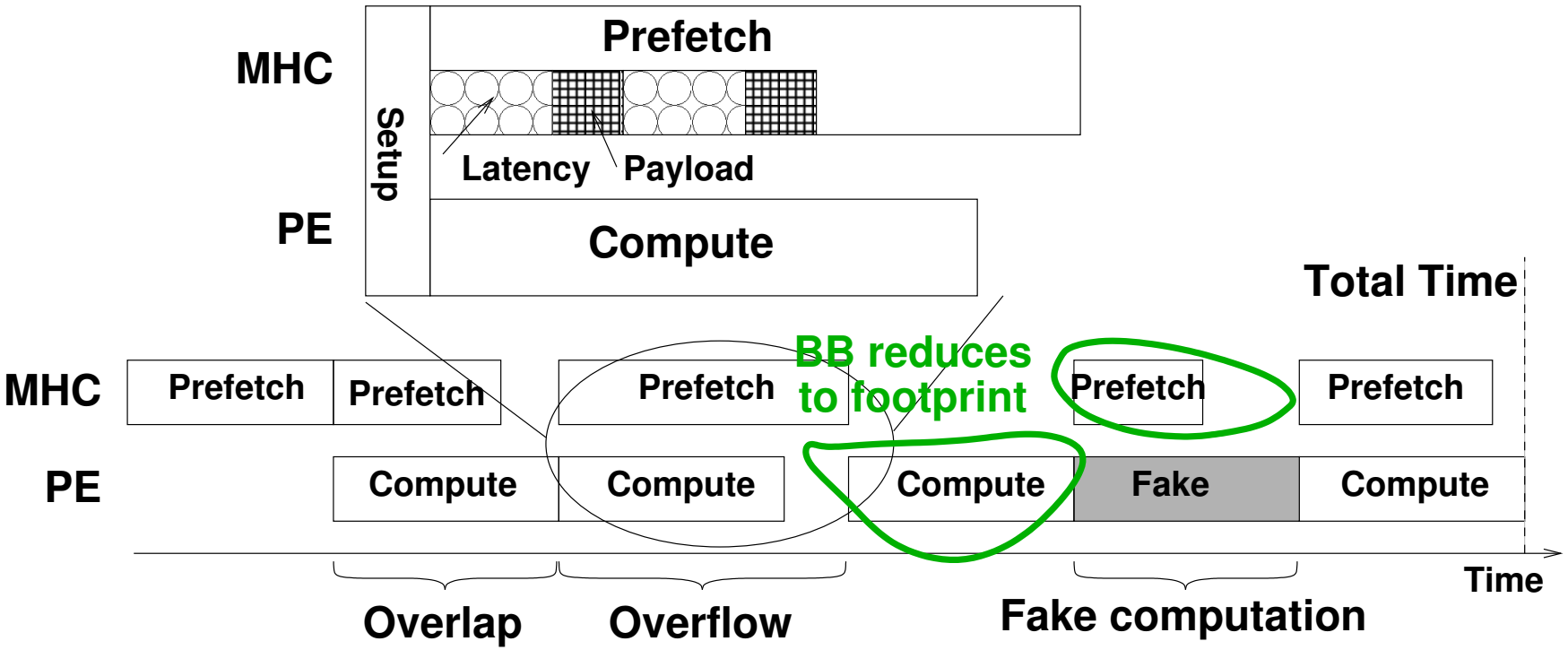
Gestion de la disparité



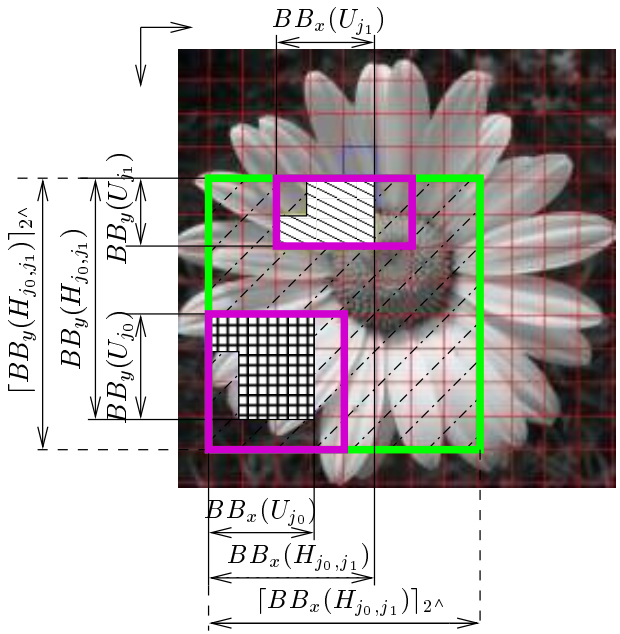
Gestion de la disparité



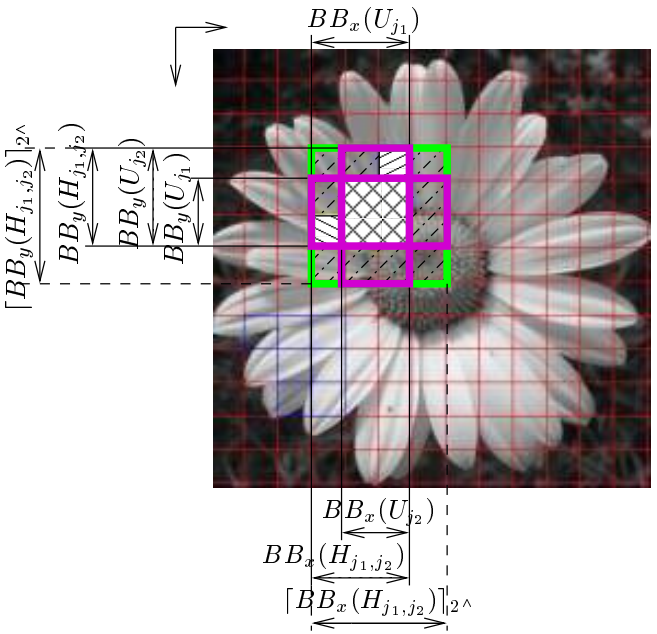
Gestion de la disparité



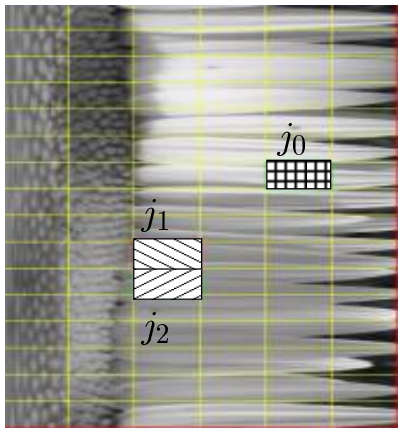
Optimisation des ressources mémoire



a) Calcul j_0 (pré-charge pour j_1)



b) Calcul j_1 (pré-charge pour j_2)

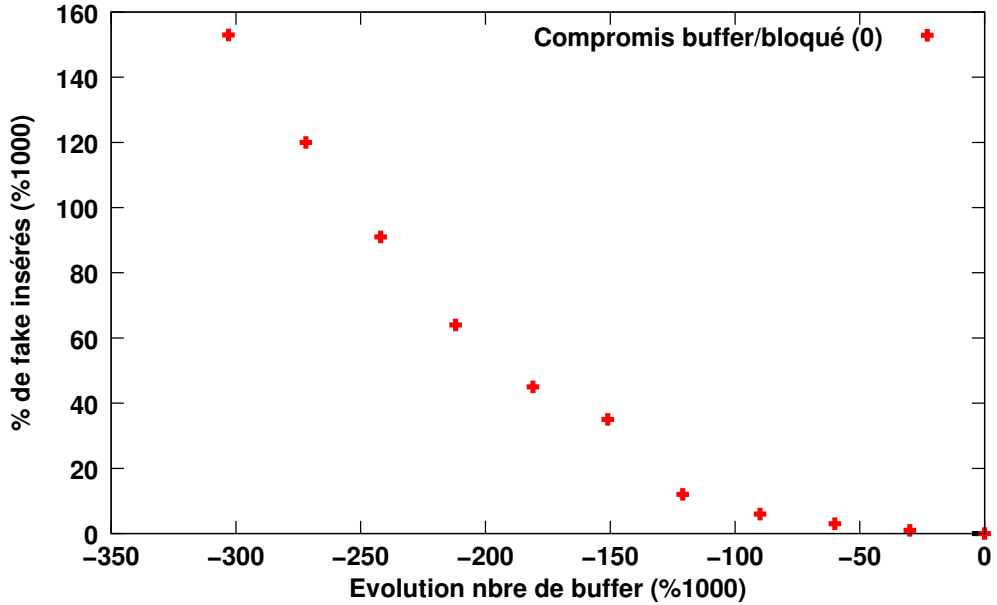


c) Sortie

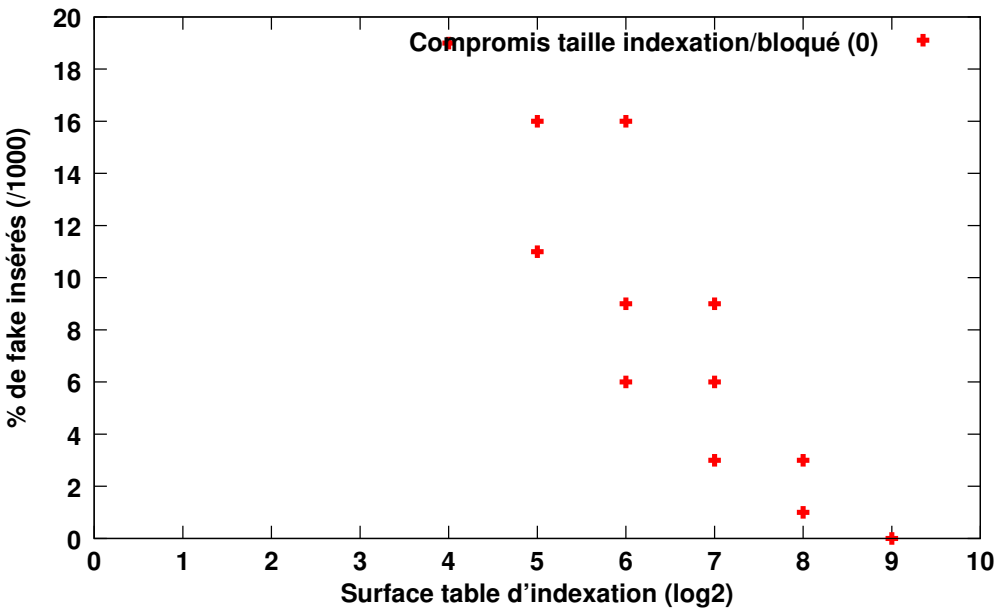
Séquencement $(j_0, \textbf{fake}, j_1, j2)$ nécessite:

- ☆ 7 buffers (au lieu de 12)
- ☆ Table IT de 4x4=16 mots (au lieu de 8x8=64 mots)

Compromis disponibles

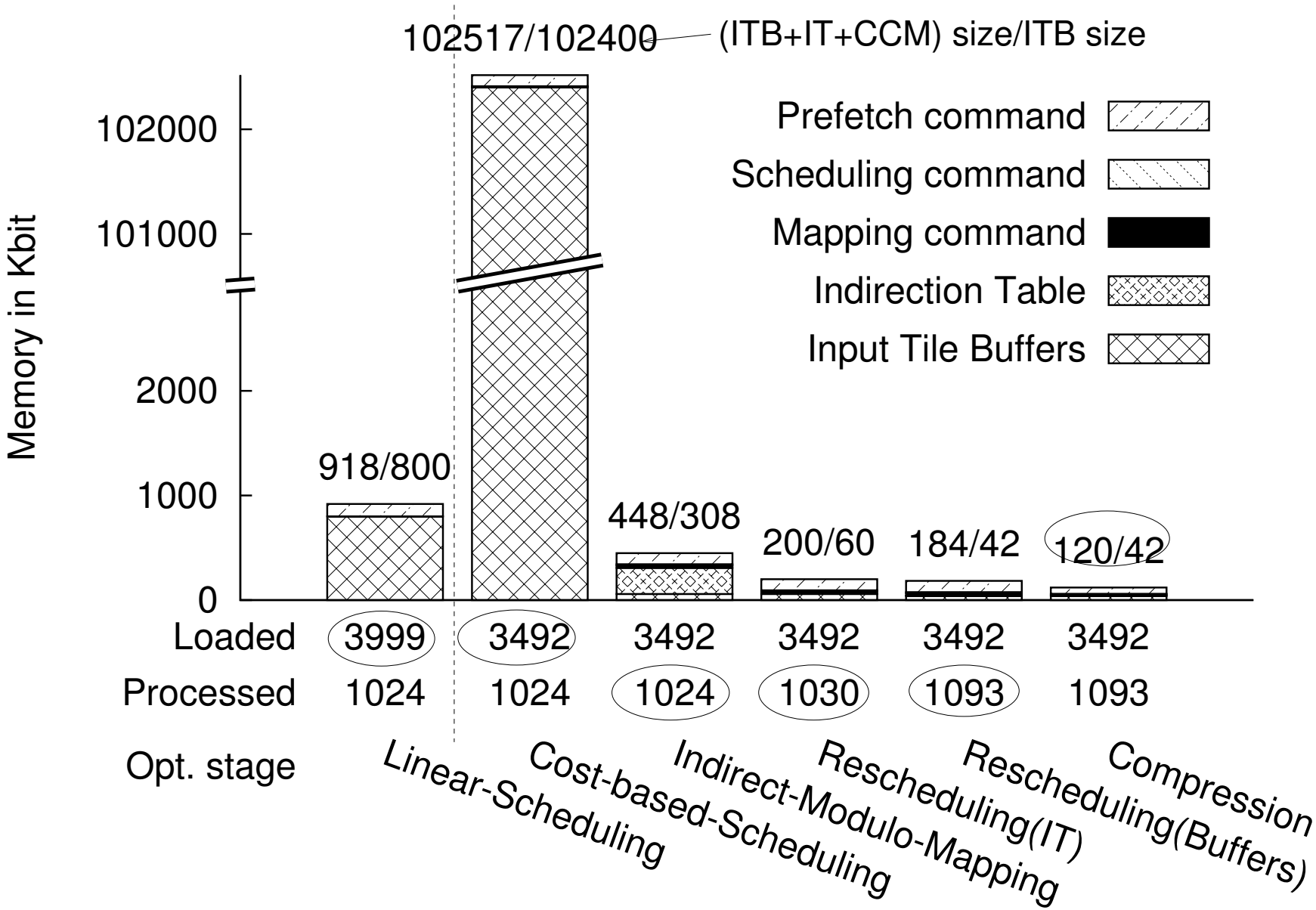


Compromis nombre de buffers/temps



Compromis surface table d'indexation/temps

Détail du processus d'optimisation (trans. polaire)



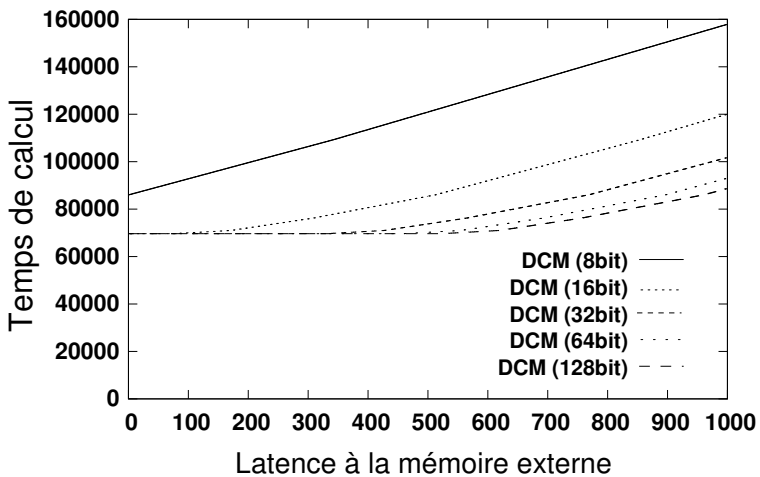
Résultats de l'optimisation

Kernel	Input data type	Input data dimension	MMopt versus			
			CatapultC	Linear scheduling		
			Memory reduction	Memory reduction (times)	Processed tile increase	Traffic saving
fisheye	image	2D	94	8	19%	29%
fisheye	mipmap anisotropic	4D	$>> 100$	12.5	19%	34%
polar	image	2D		12	21%	36%
polar	mipmap anisotropic	4D		27	21%	24%
polar	mipmap anisotropic	2D (flat)		51.5	20%	92%
polar	mipmap isotropic	3D		7	02%	38%
polar	mipmap isotropic	2D (flat)		305	02%	38%
pseudolog	image	2D		12	20%	19%
pseudolog	mipmap anisotropic	4D		34	14%	23%
pseudolog	mipmap anisotropic	2D (flat)		124	09%	37%
Average				**32**	**15%**	**30%**

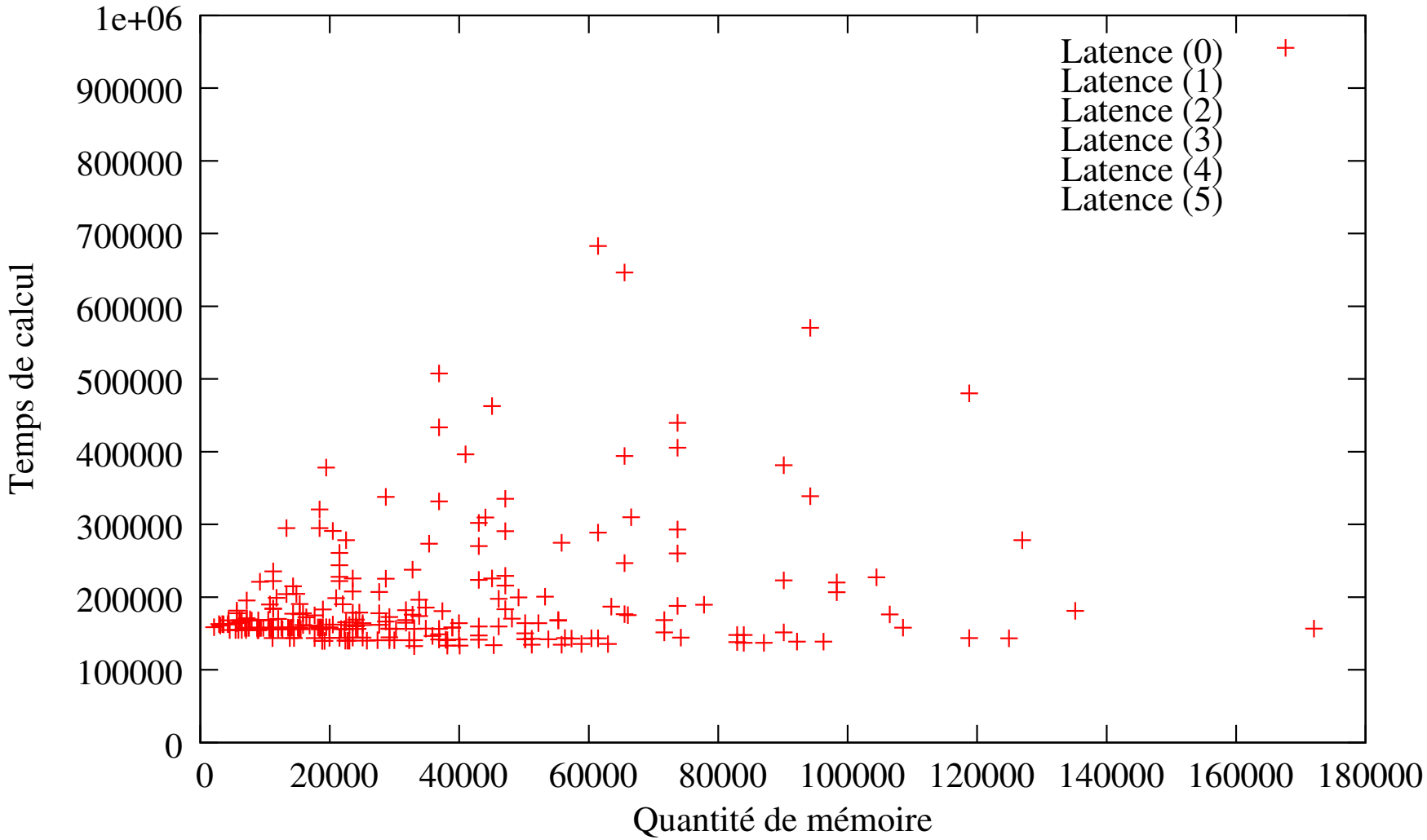
Résultats de synthèse (trans. polaire)

	LUT	CLB	DFF	Pipeline (profondeur)
PE (Sans DCM)	278	139	262	8
PE+DCM (8bit)	525	282	525	11
sur-coût	247	143	263	3
PE+DCM (32bit)	631	316	587	11
sur-coût	353	177	325	3
PE+DCM (64bit)	772	387	680	11
sur-coût	494	248	418	3

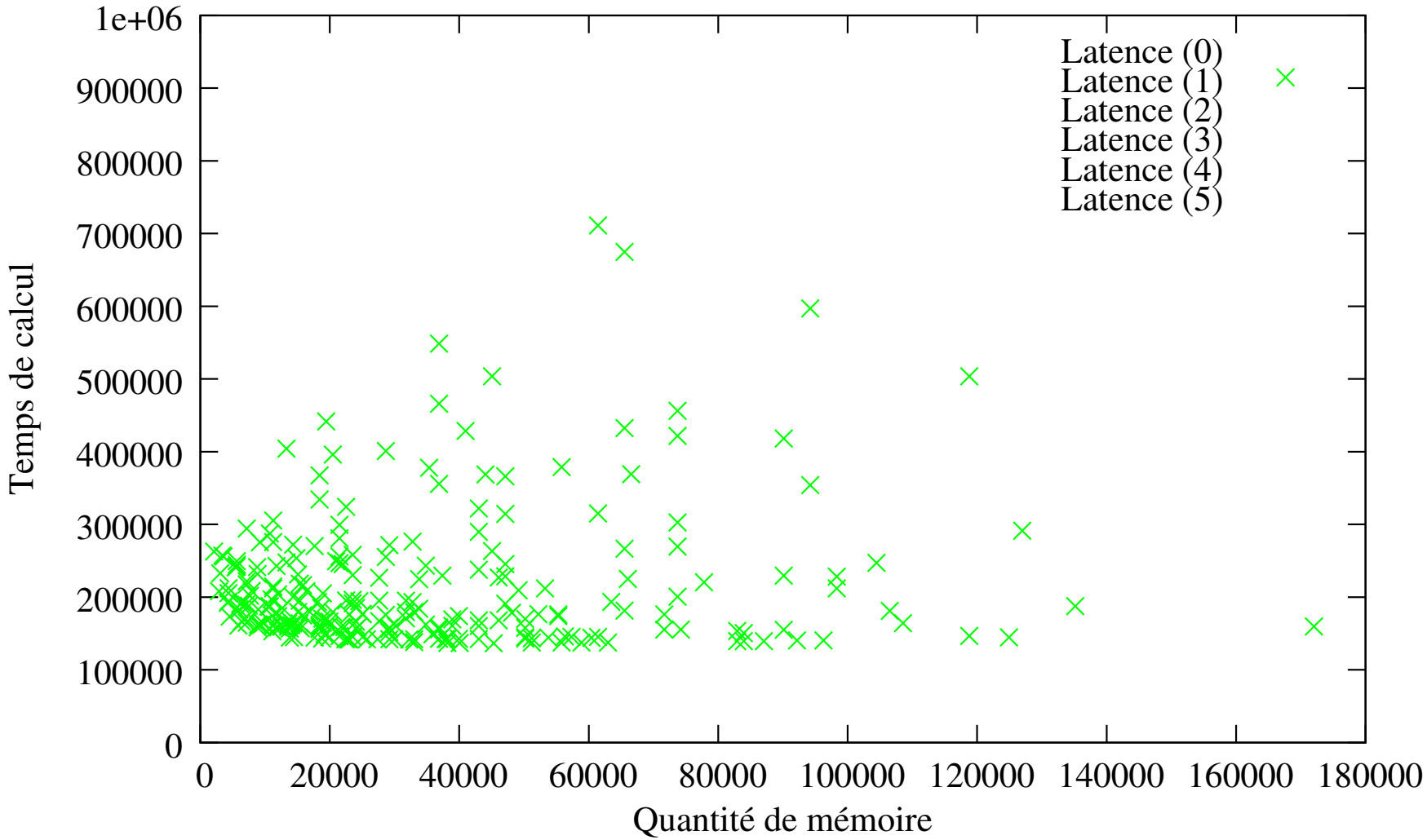
Ne tient pas compte des DSP48 et ROMs pour opérations arithmétiques !



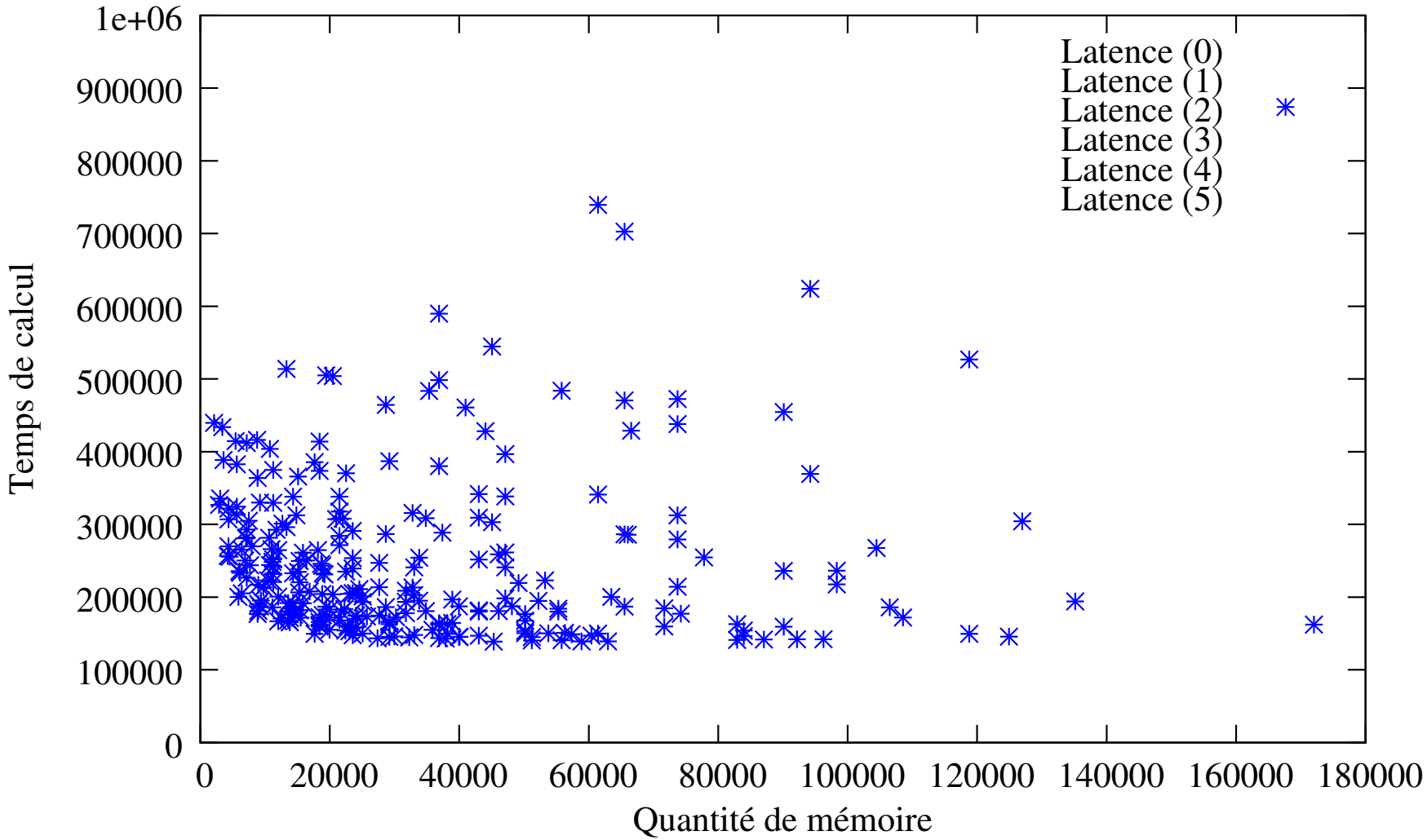
Choisir un tuilage ?



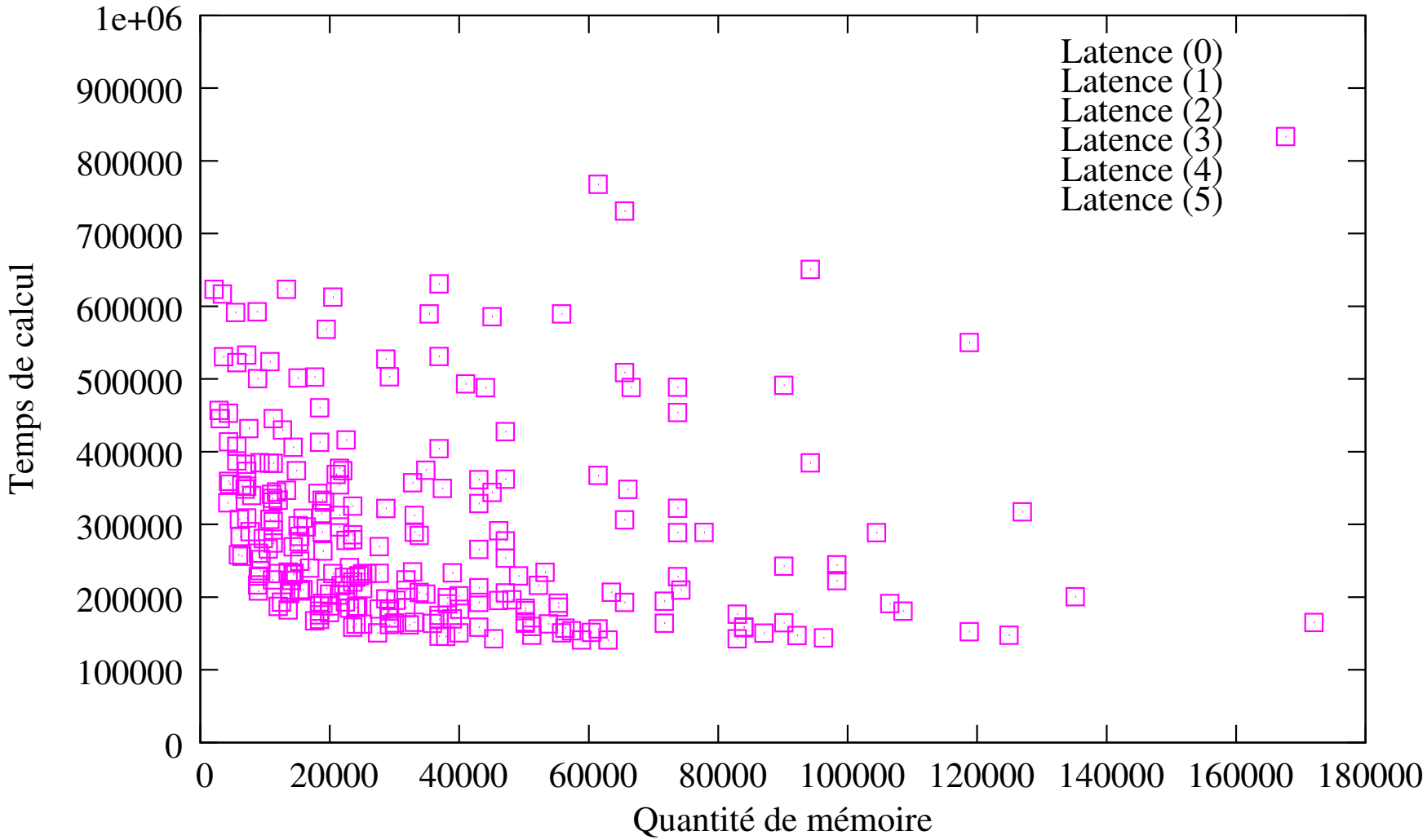
Choisir un tuilage ?



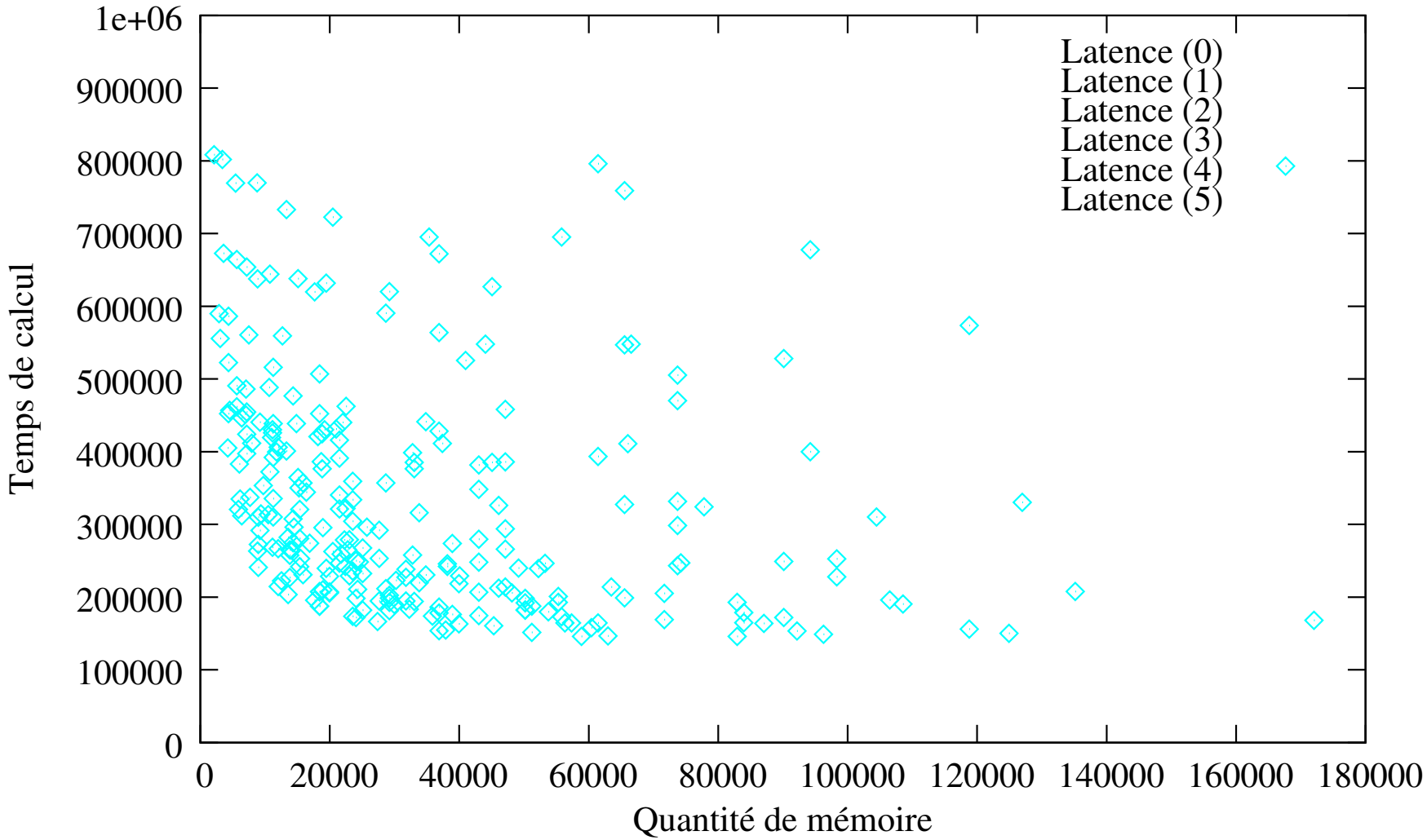
Choisir un tuilage ?



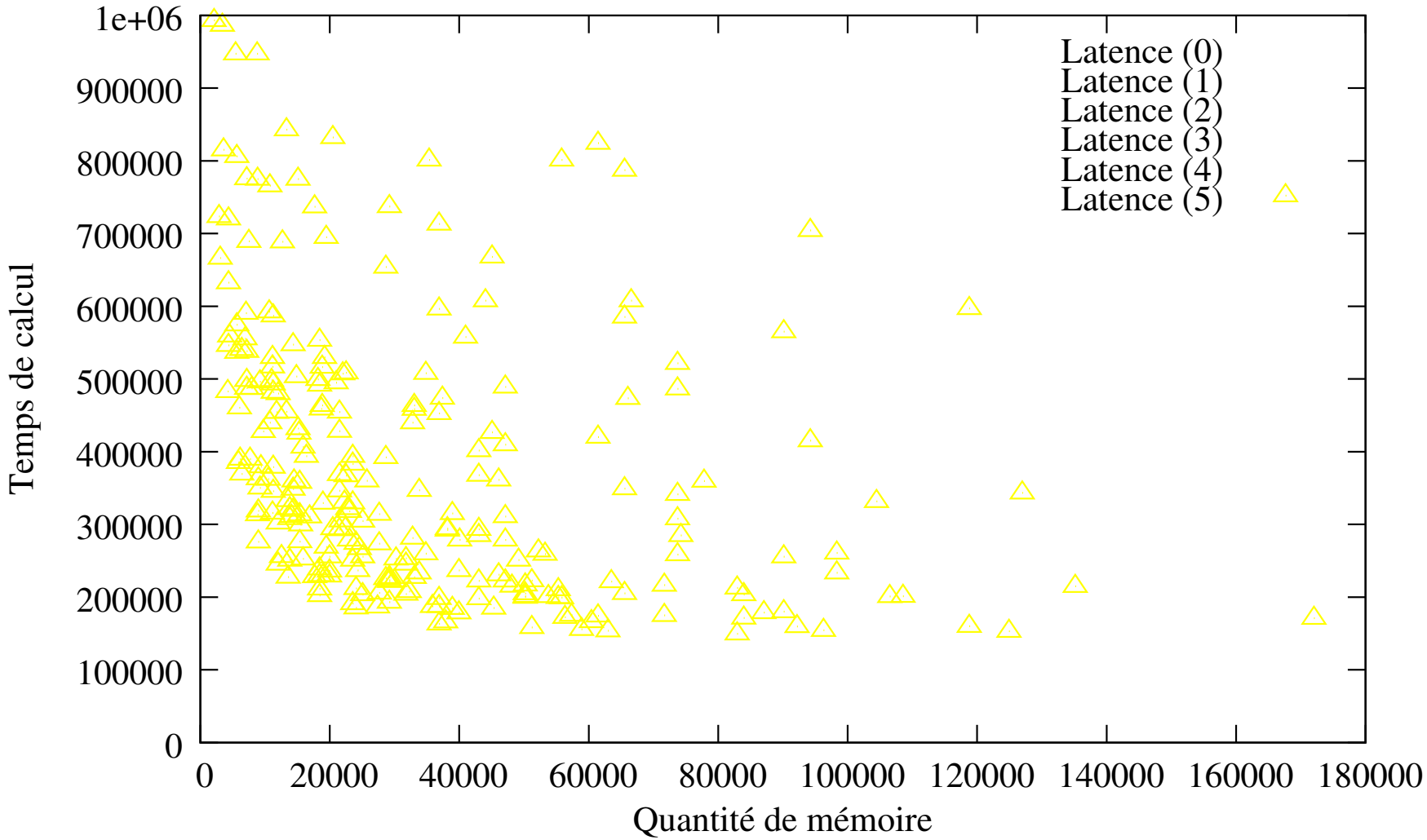
Choisir un tuilage ?



Choisir un tuilage ?



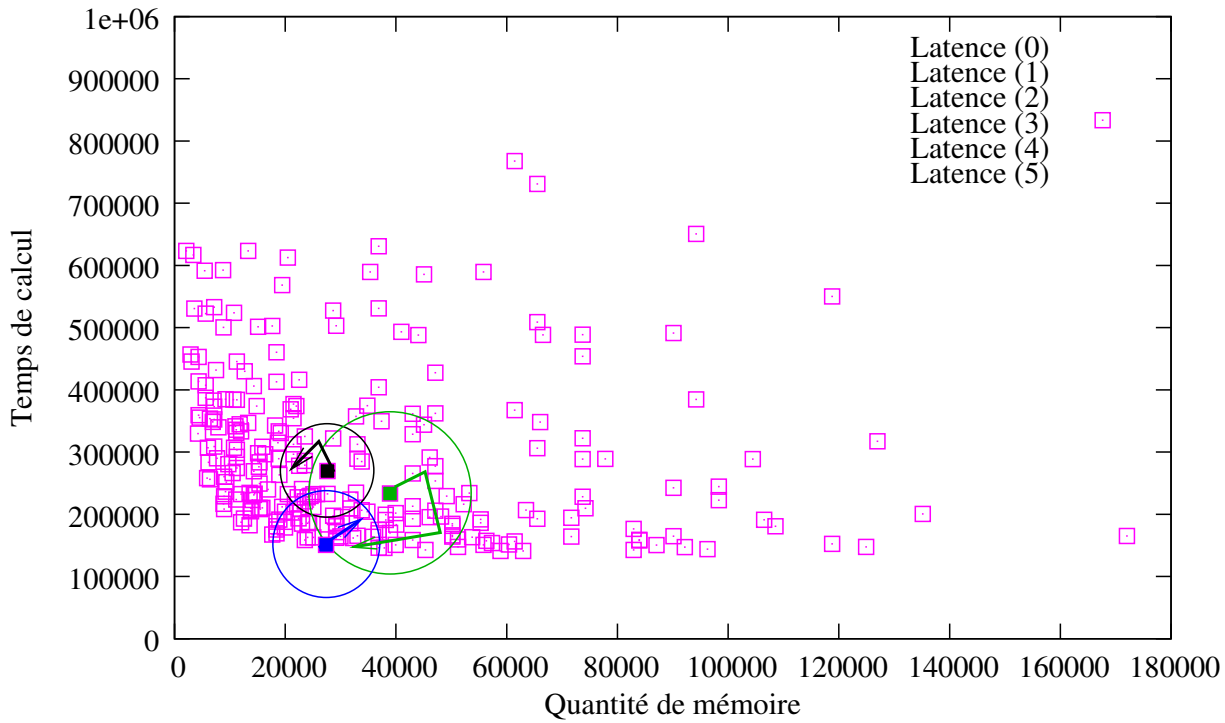
Choisir un tuilage ?



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?



Perspectives

- Améliorer les optimisations
- Intégrer une analyse de code dans un framework
- Passer au multi-kernel

Conclusion

- Travailler en amont de la HLS offre de nombreuses opportunités

- Méthodes globales (RO) à comparer aux méthodes “locales”, en ligne

Journée SOC-SIP 15-11-2012

Gestion des données pour les noyaux non-linéaires dans un flot de conception HLS

S. Mancini

Plan Détaillé