WASC 2012

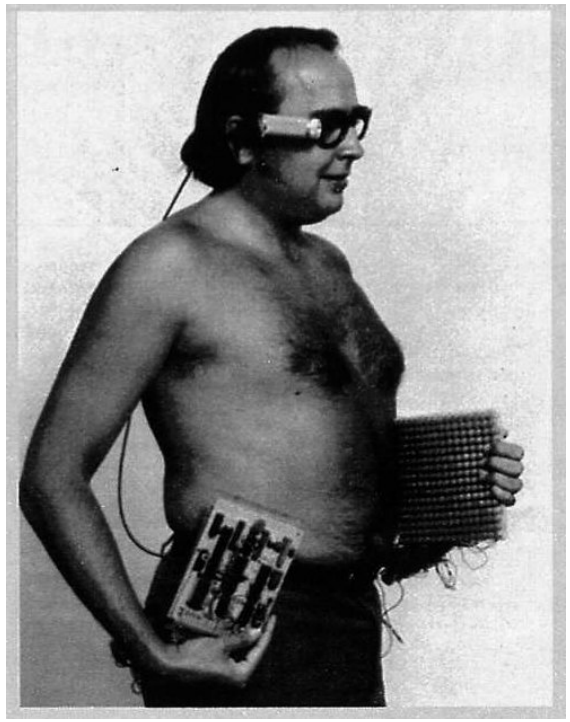
Design of non-linear kernel IPs for vision systems

S. Mancini & F. Rousseau

Context

Embedded systems for image processing

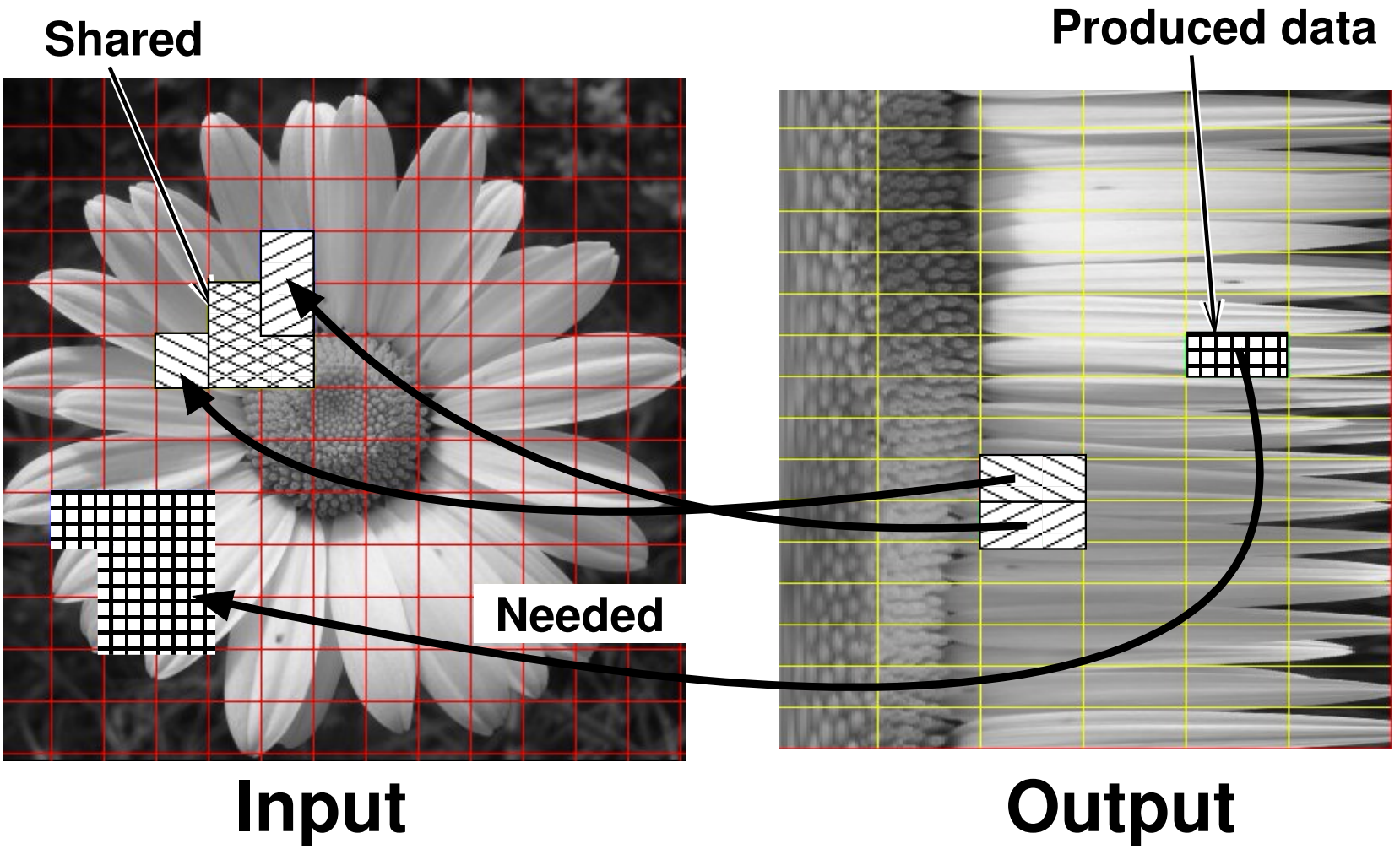
- Specificities
  - “Low level” processings
  - High definition images
- Architectural constraints
  - Limited resources
  - Low power
  - Real time



The data transport is becoming the bottleneck due to the memory wall :

⇨ Need to adapt the memory hierarchy and data control

Non-linear kernels



Problems

- The size of the data requires the use of large low cost external memories (xDDR-SDRAM)
- The memory wall is worsen :
  - Acces times are “constant” but computing units are getting faster
  - The bandwith is shared by several processing elements

  ⇨ Transactions to memories are getting slower relatively to each computing unit
- There’s a need to store local copies of data

  ⇨ Their management is complex for non-linear kernels because memory references are irregular

Problems II

We would like to design IPs with HLS frameworks but :

- Standard HLS flows do not manage the data transport to external memories
- The HLS tools are efficient to generate architectures for which the data are present in local memories
- HLS tools optimizations are based on regular loops

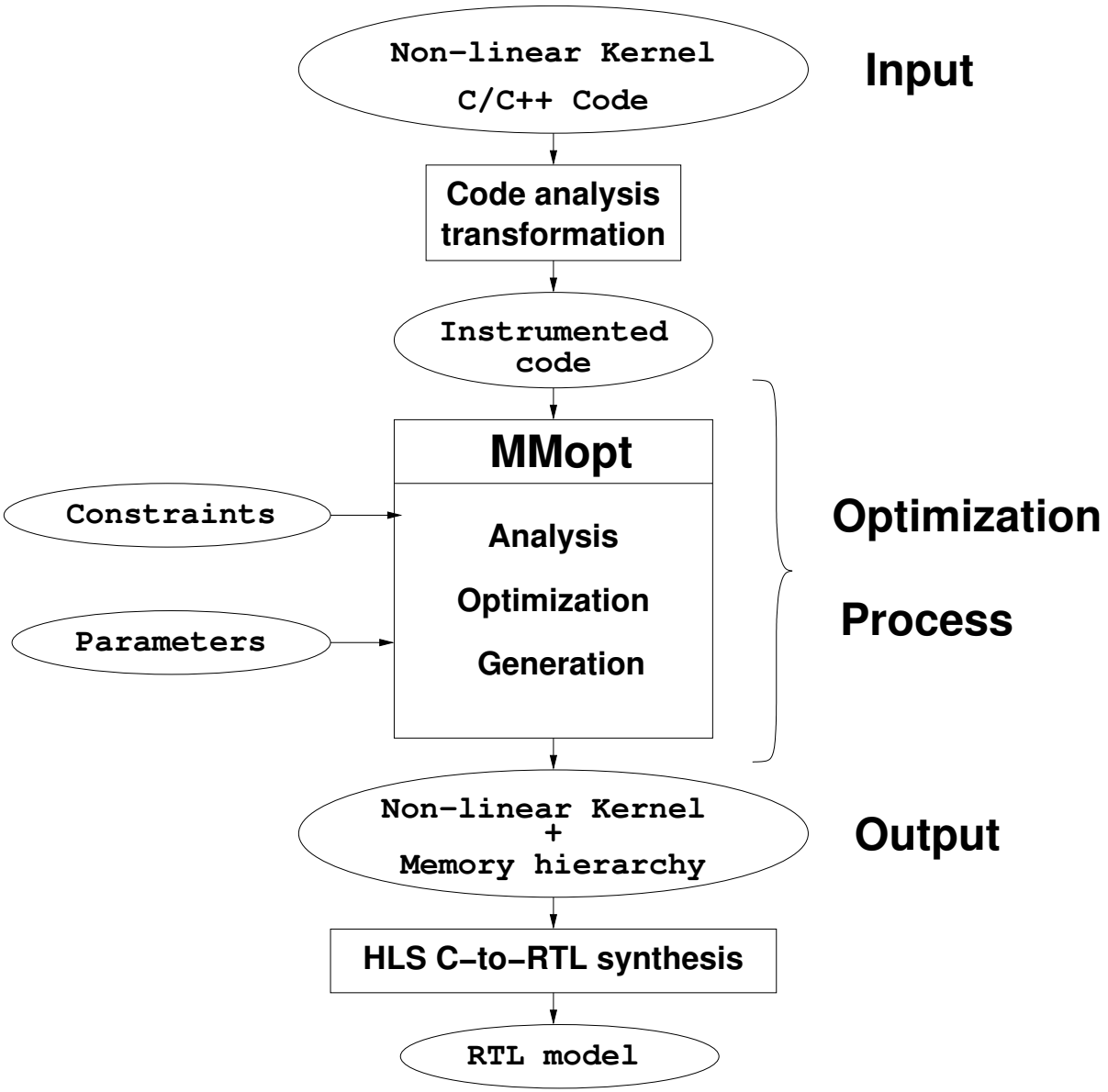
Goals

Optimize the management of the data in a HLS flow.

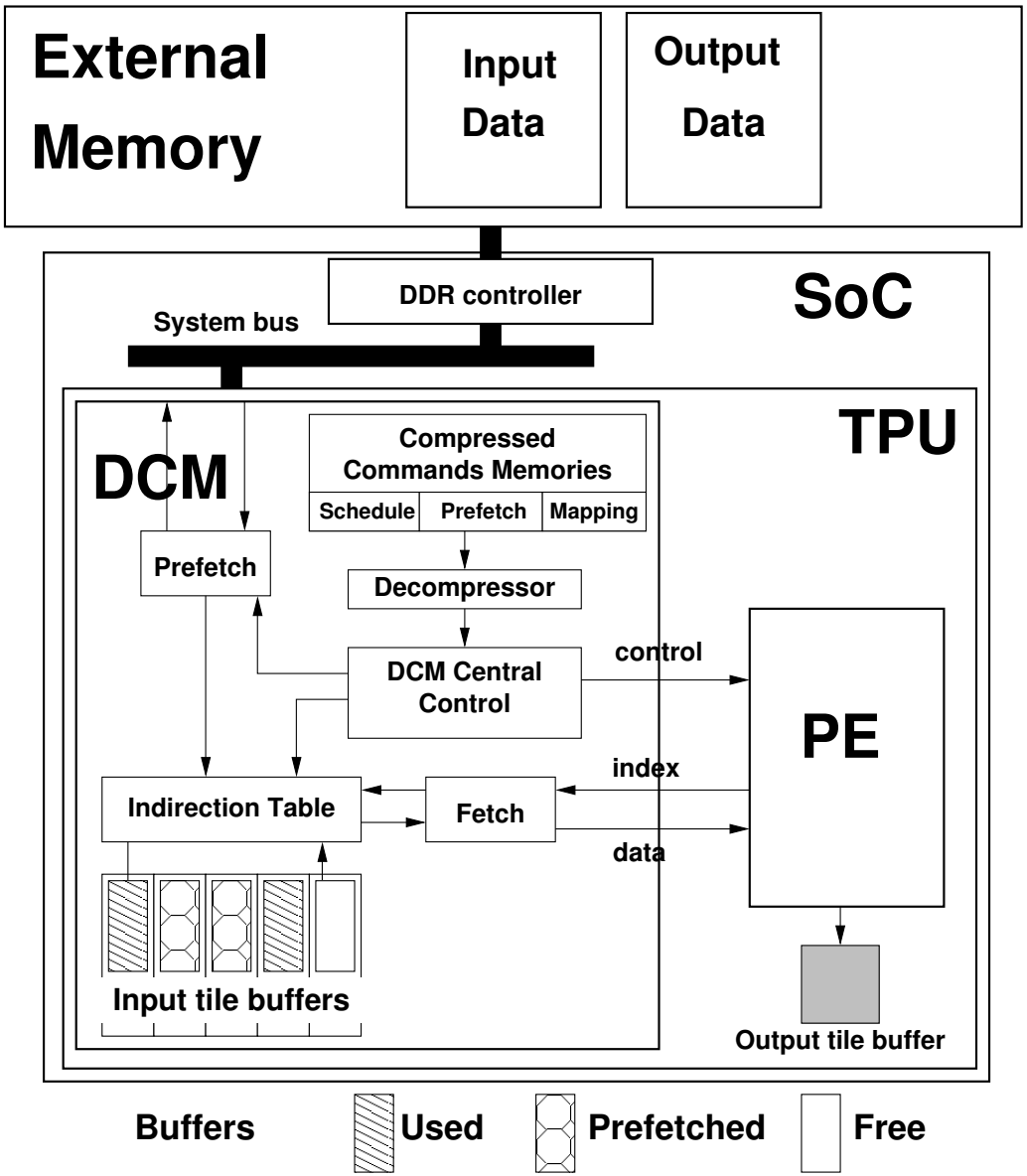
Ideas :

- Start from a high level code for HLS and enhance it with a data controller
- The obtained design should comply with HLS frameworks
- Manage the disparity of non-linear kernels before HLS

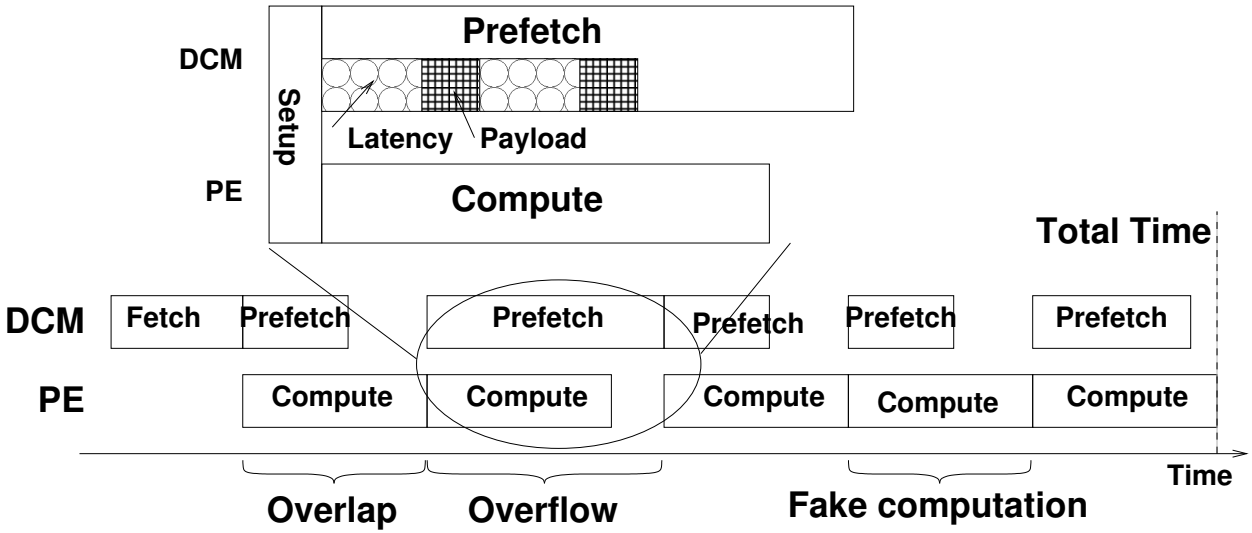
The proposed method

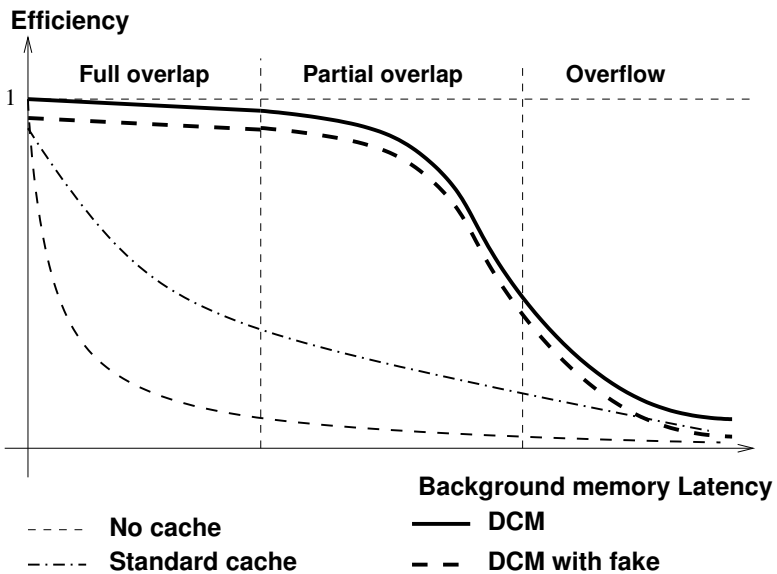


Target architecture



Scheduling


DCM
Prefetch
Setup
Latency
Payload
PE
Compute
Total Time
DCM
Fetch
Prefetch
Prefetch
Prefetch
Prefetch
Prefetch
PE
Compute
Compute
Compute
Compute
Compute
Time
Overlap
Overflow
Fake computation

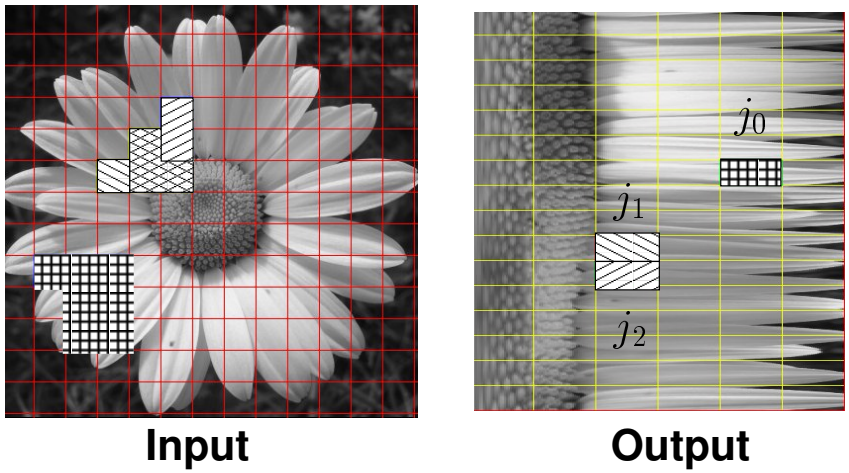

Efficiency
Full overlap
Partial overlap
Overflow
1
Background memory Latency
No cache
Standard cache
DCM
DCM with fake

Optimizations

There's a need to optimize :

- The quantity of data copied from the external memory
- The size of embedded memories
  - The size and number of buffers
  - The size of "control" memories
    - Indirection Table
    - Command memories (scheduling, mapping, pre-fetch)

Traffic minimisation



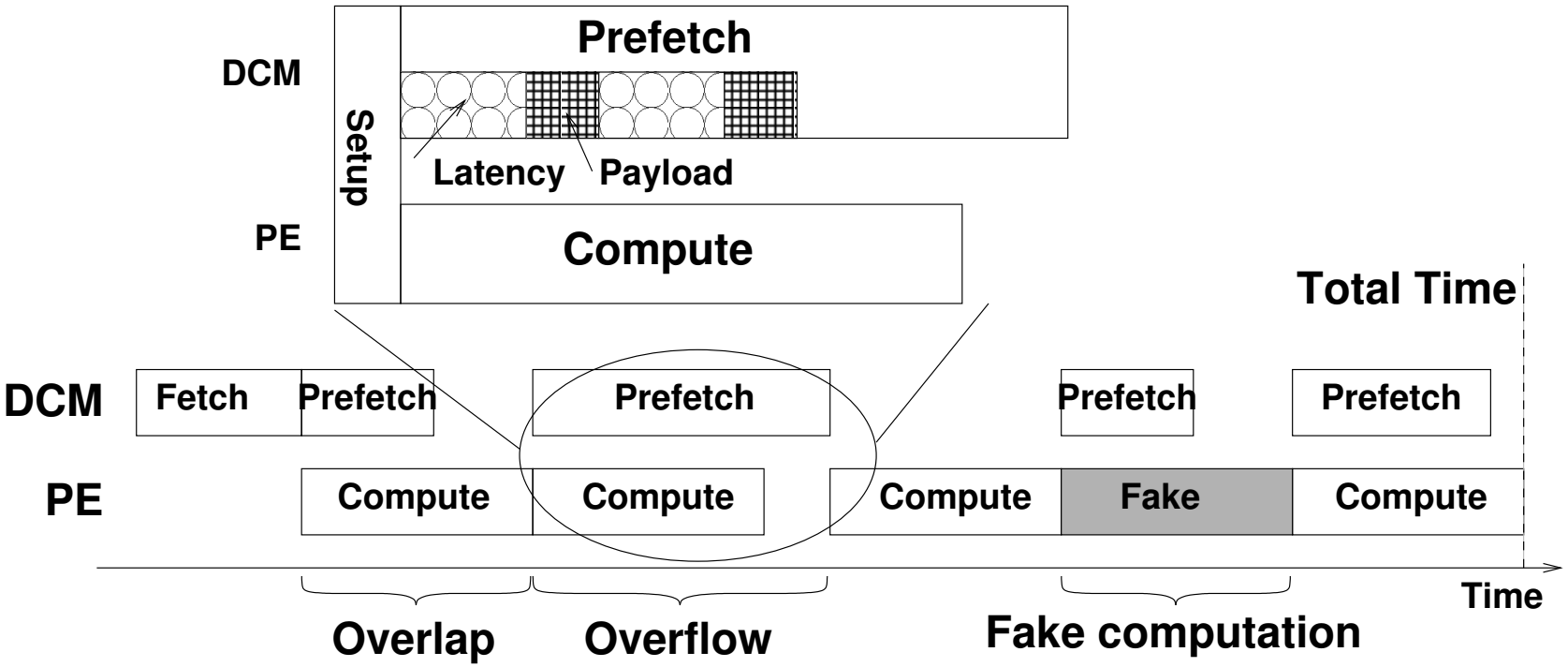
Input **Output**

The number of loaded tiles depends on the schedule of the output computations. A “first order” approximation, with no liveliness optimization, gives :

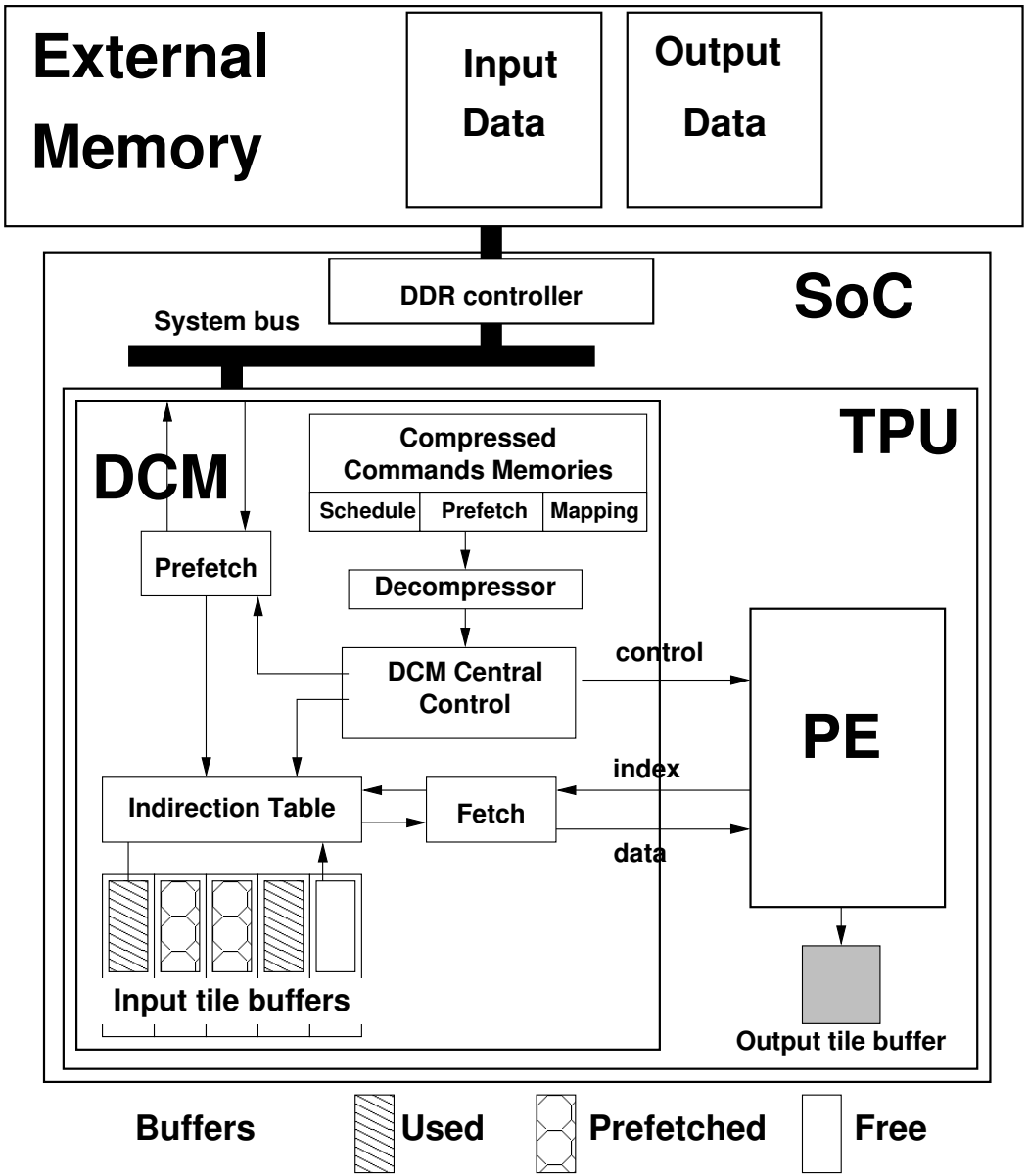
- j_1, j_0, j_2 needs to load 15 tiles
- j_0, j_1, j_2 needs to load 12 tiles

A basic “first order” scheduling is equivalent to the ATSP problem : it is NP-complex !
Minimising the traffic may increase the memory resources.

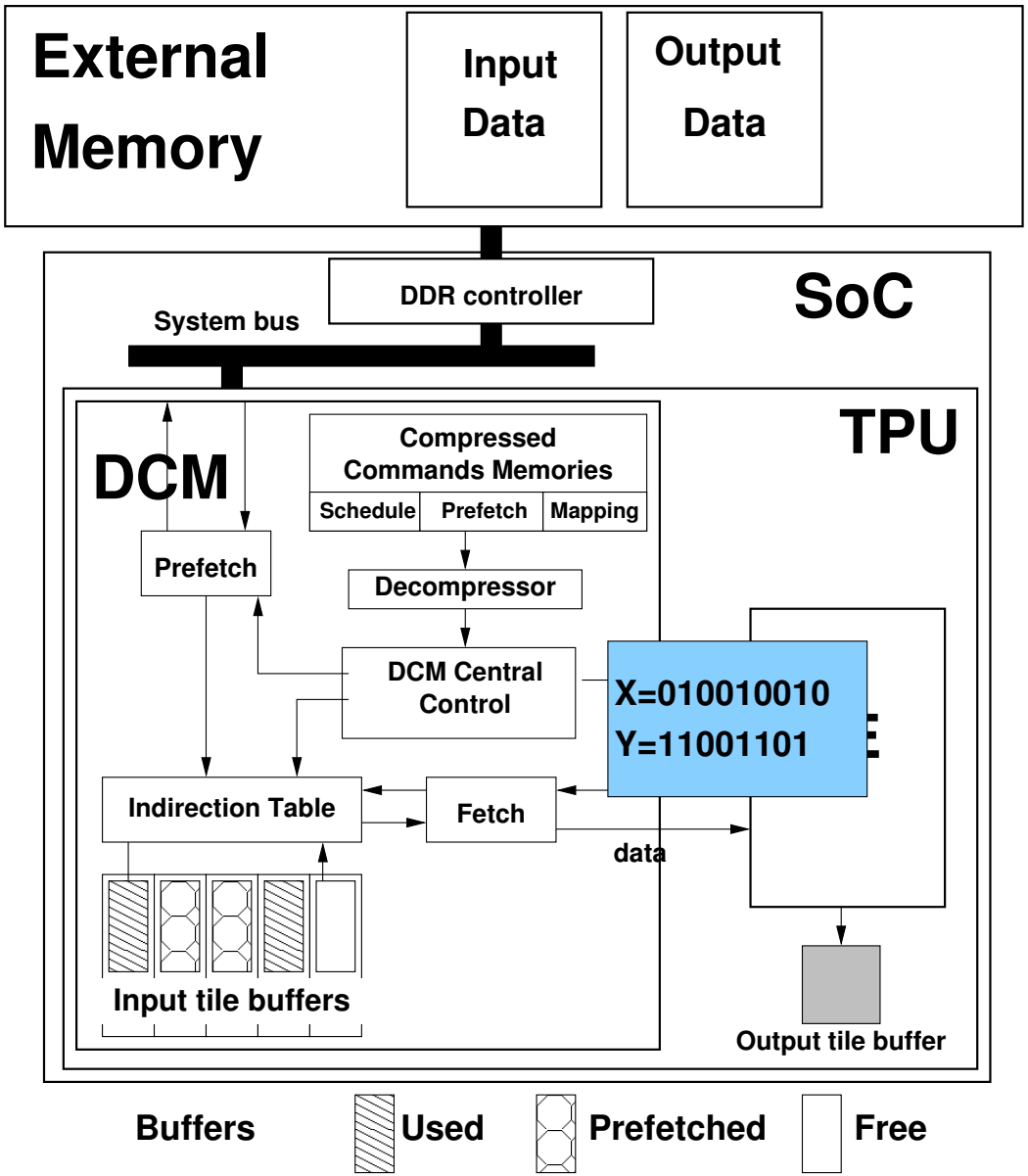
Managing the disparity by breaking the regularity



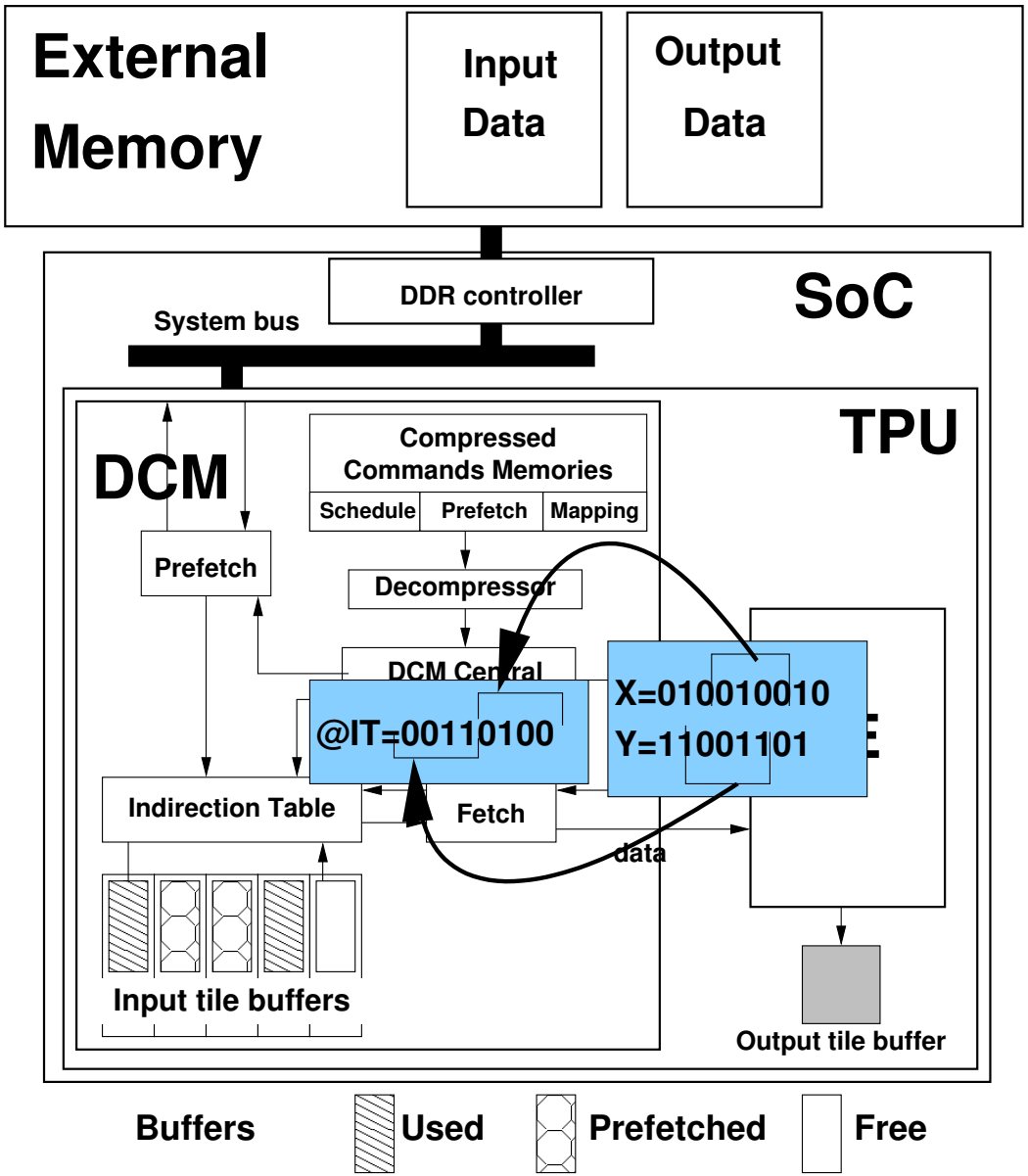
IT optimization



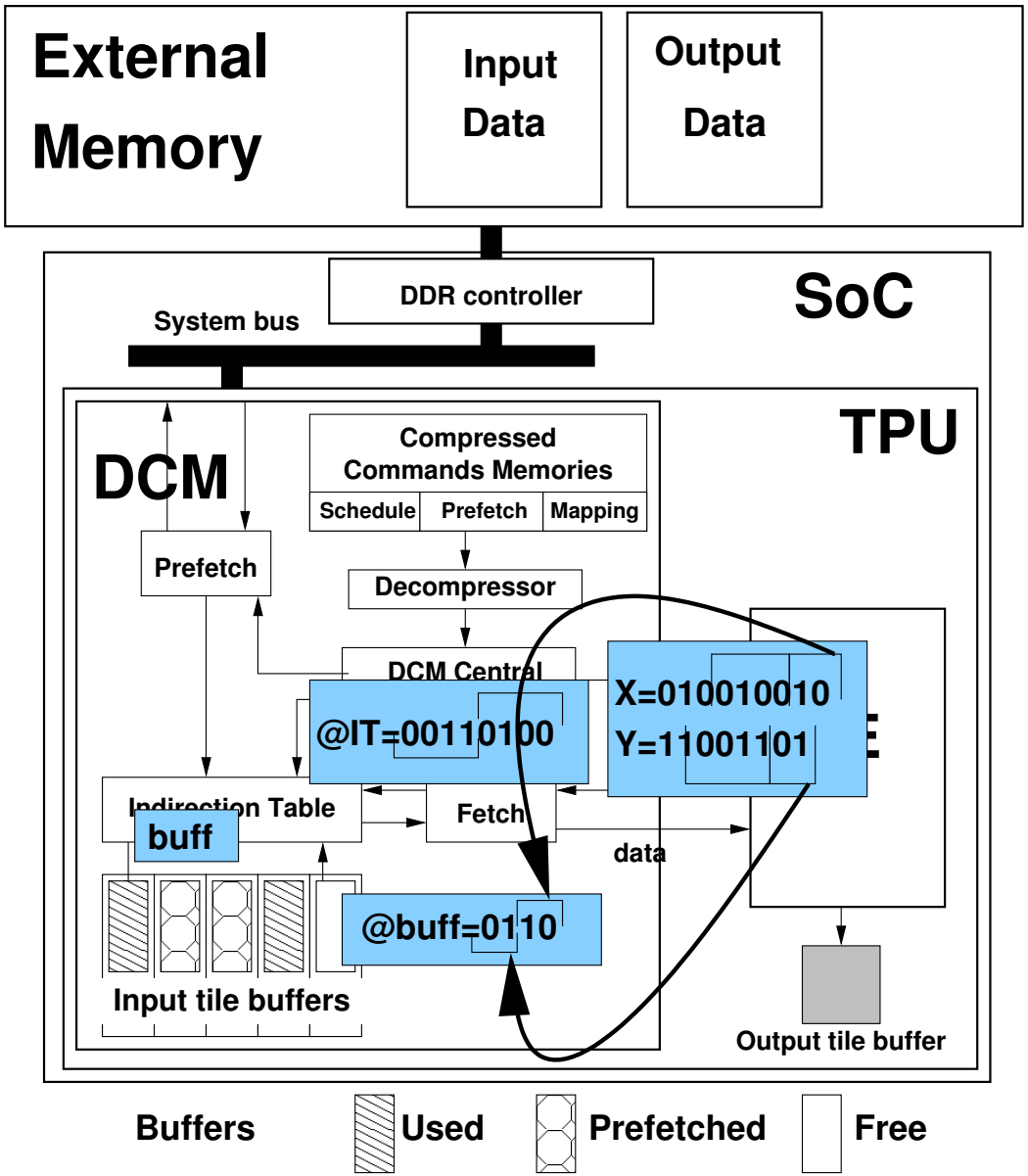
IT optimization



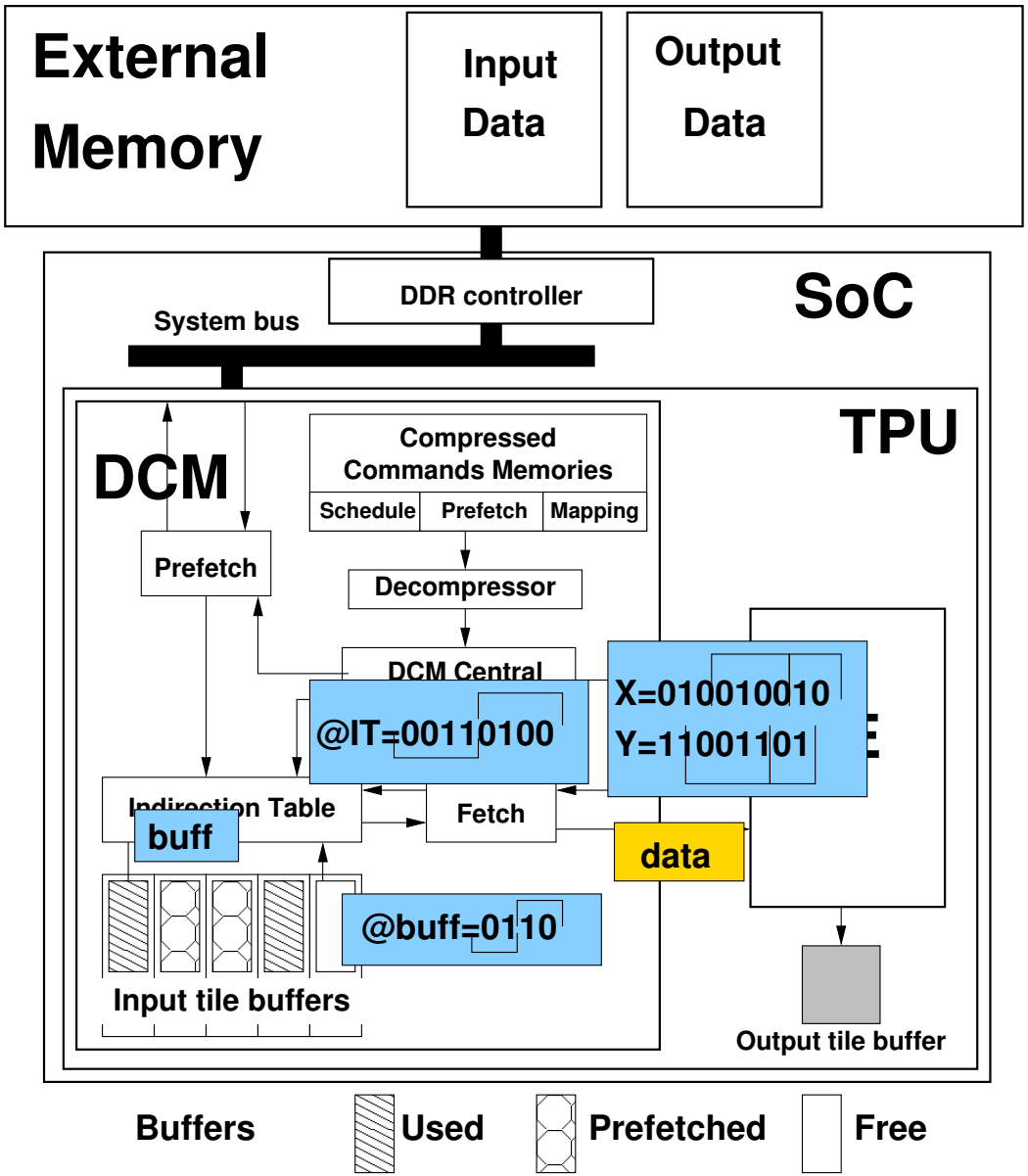
IT optimization



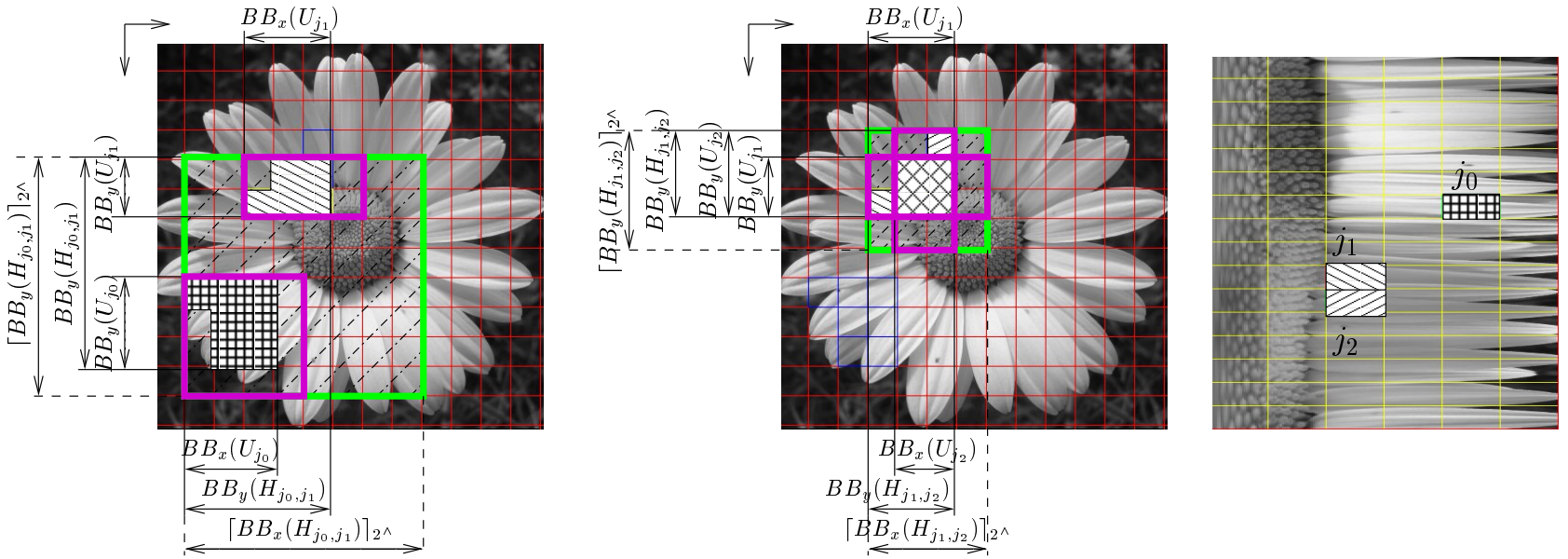
IT optimization



IT optimization



Computing the IT size



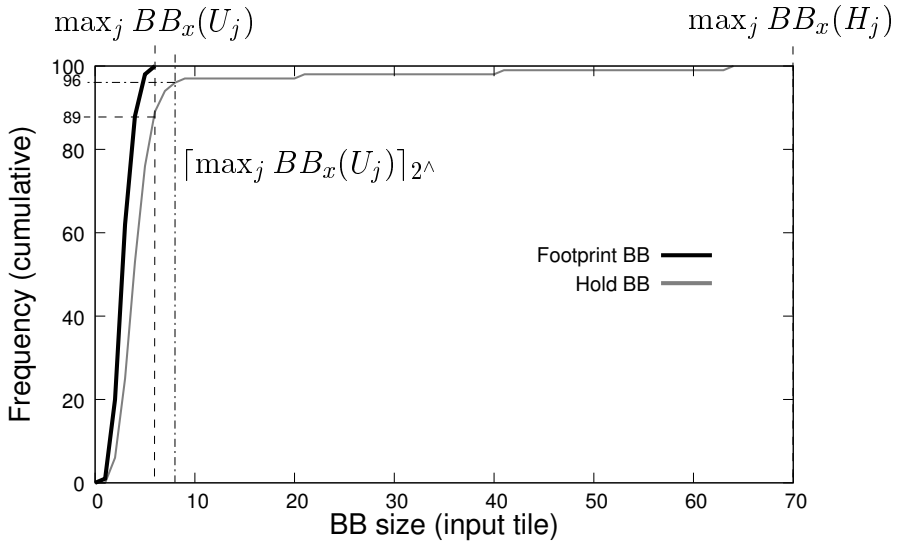
a) Step j_0 (prefetch j_1) b) Step j_1 (prefetch j_2) c) Output

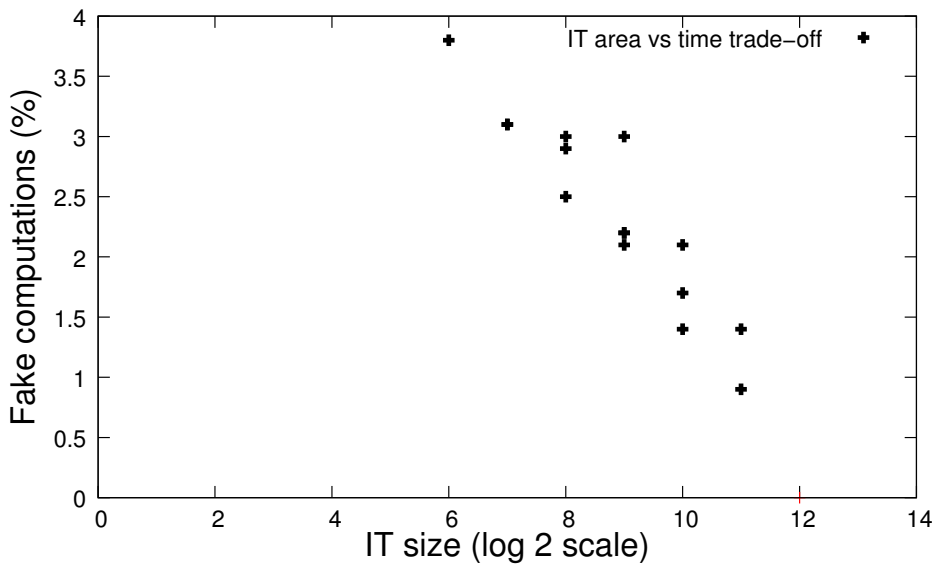
The IT size depends on the schedule s and is defined as :

$$m_s = \prod_d \max_i \lceil BB_d(H_{s_i}, s_{i+1}) \rceil_{2^\wedge}$$

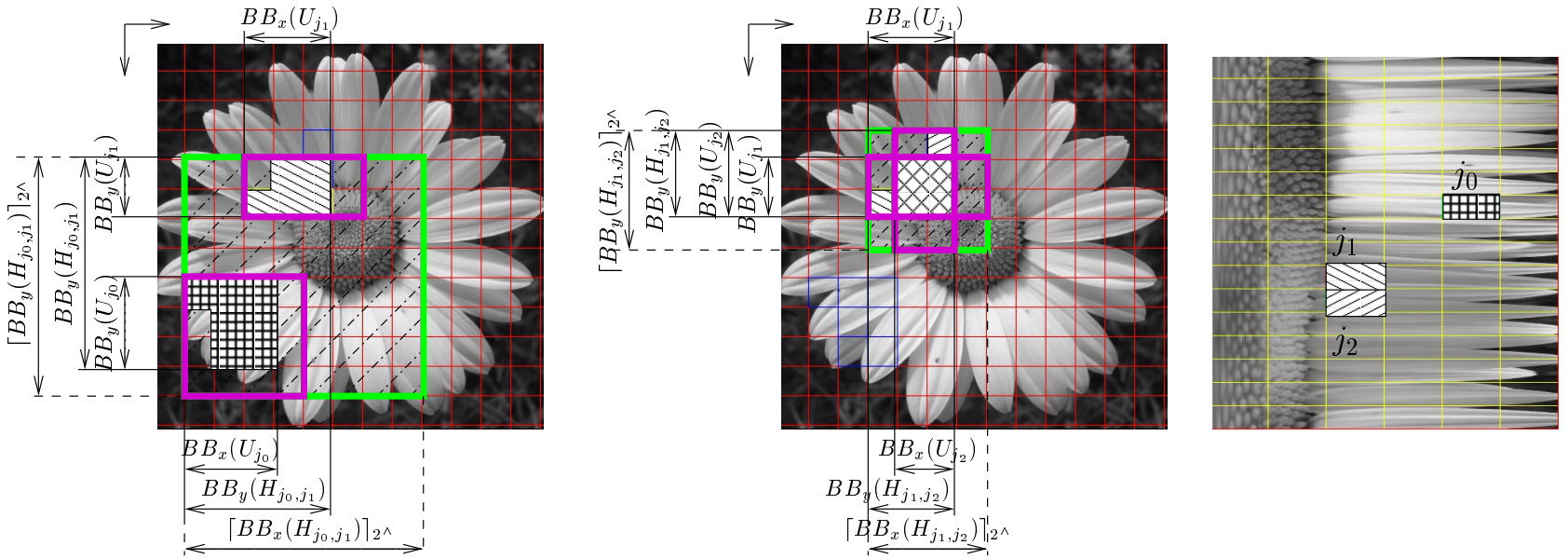
- The schedule $\{j_0, j_1, j_2\}$ induces $m = 64$
- The schedule $\{j_0, \textbf{fake}, j_1, j_2\}$ induces $m = 16$

Trade-offs IT vs time





Buffer optimization



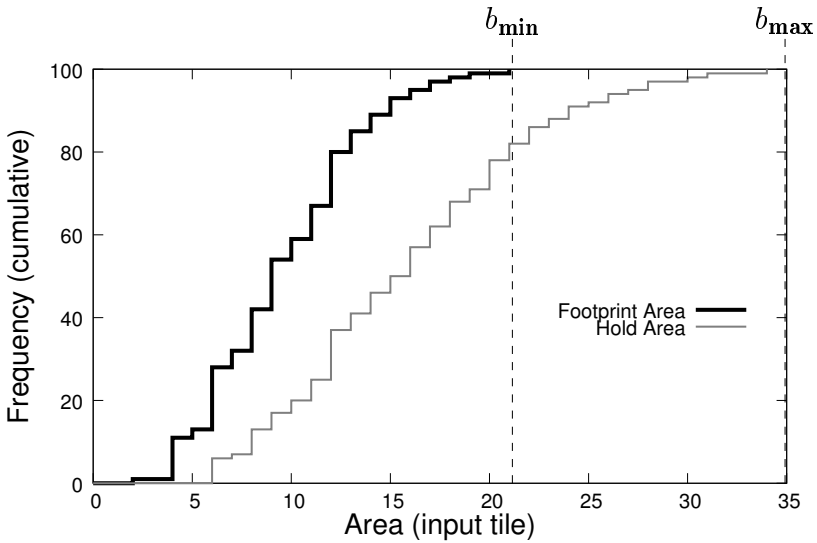
a) Step j_0 (prefetch j_1) b) Step j_1 (prefetch j_2) c) Output

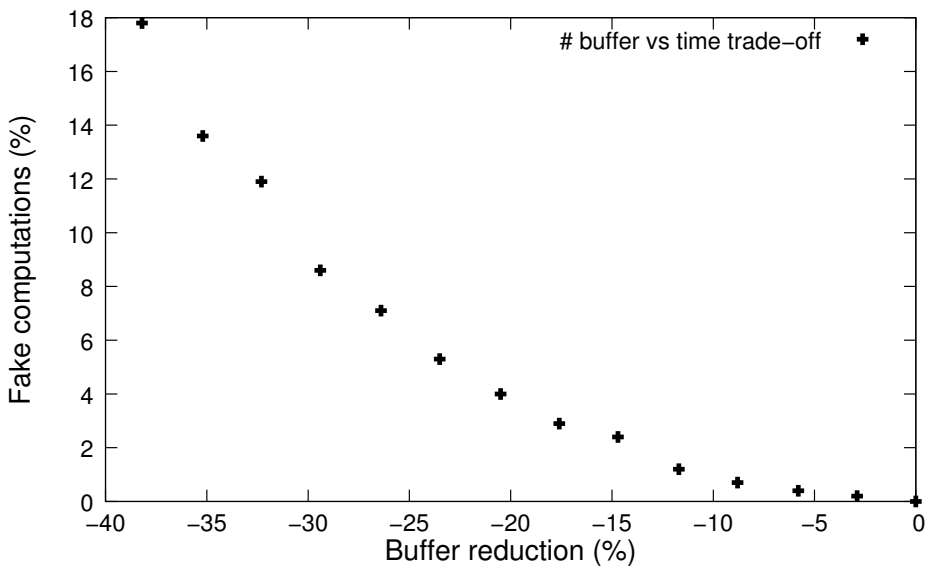
The number of buffers is defined as

$$b_s = \max_i \left| H_{s_i, s_{i+1}} \right|$$

- The schedule $\{j_0, j_1, j_2\}$ induces $b = 12$
- The schedule $\{j_0, \textbf{fake}, j_1, j_2\}$ induces $b = 7$

Trade-offs buffers vs time



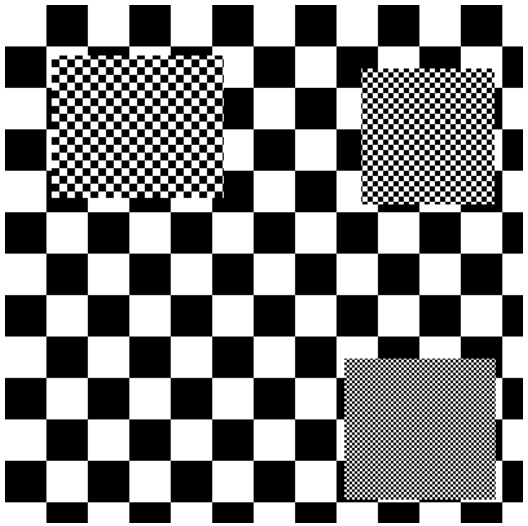


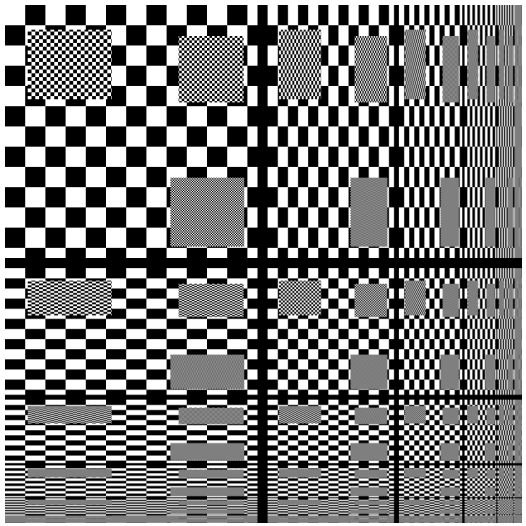
Benchmarks

- Kernels :
  - Pseudo-log sampling
  - Polar transform
  - Fisheye correction (wide angle)
- Input data :
  - Image
  - MipMap anisotropic
  - MipMap isotropic

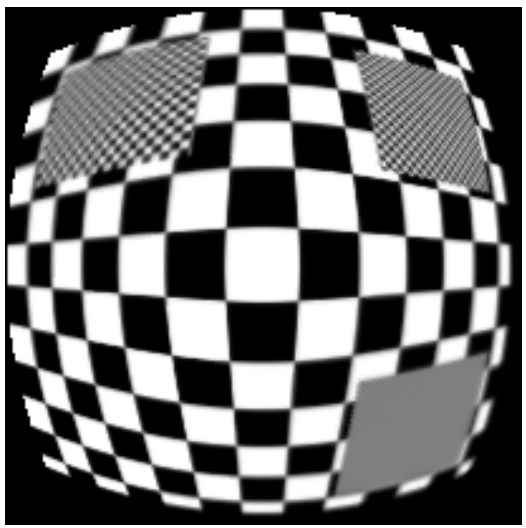
Mipmap ?

Input





Output



Results

Kernel	Input data type	Input data dimension	MMopt versus CatapultC: Memory reduction	MMopt versus Linear scheduling: Memory reduction (times)	MMopt versus Linear scheduling: Processed tile increase	MMopt versus Linear scheduling: 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%**

Conclusion & perspectives

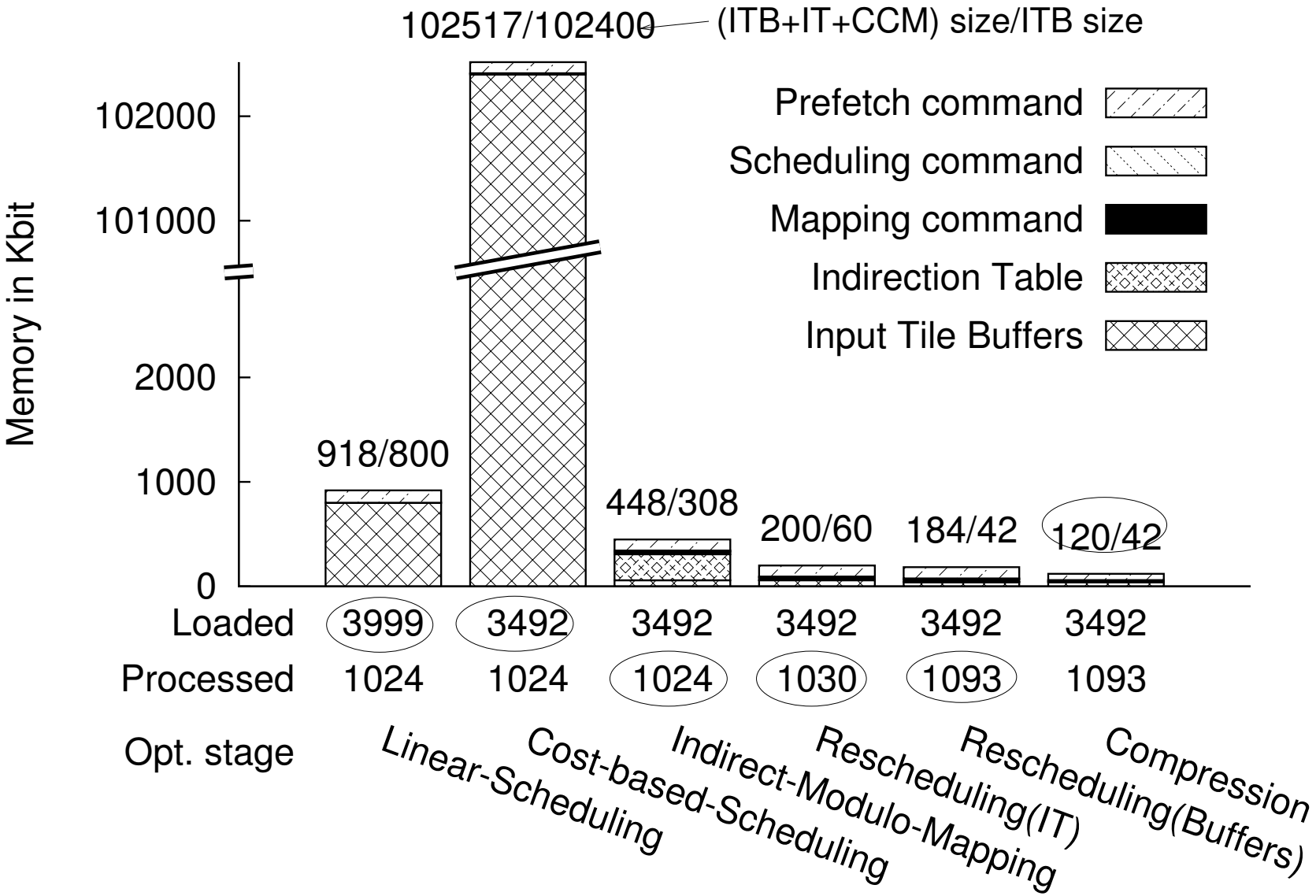
Conclusion :

- The MMopt optimization tools is efficient and enables several trade-offs
- The tool generates the Data and Memory Controller configuration and commands
- Integrating the management of data prior to HLS is of high added value
- The design of image processing kernels is faster

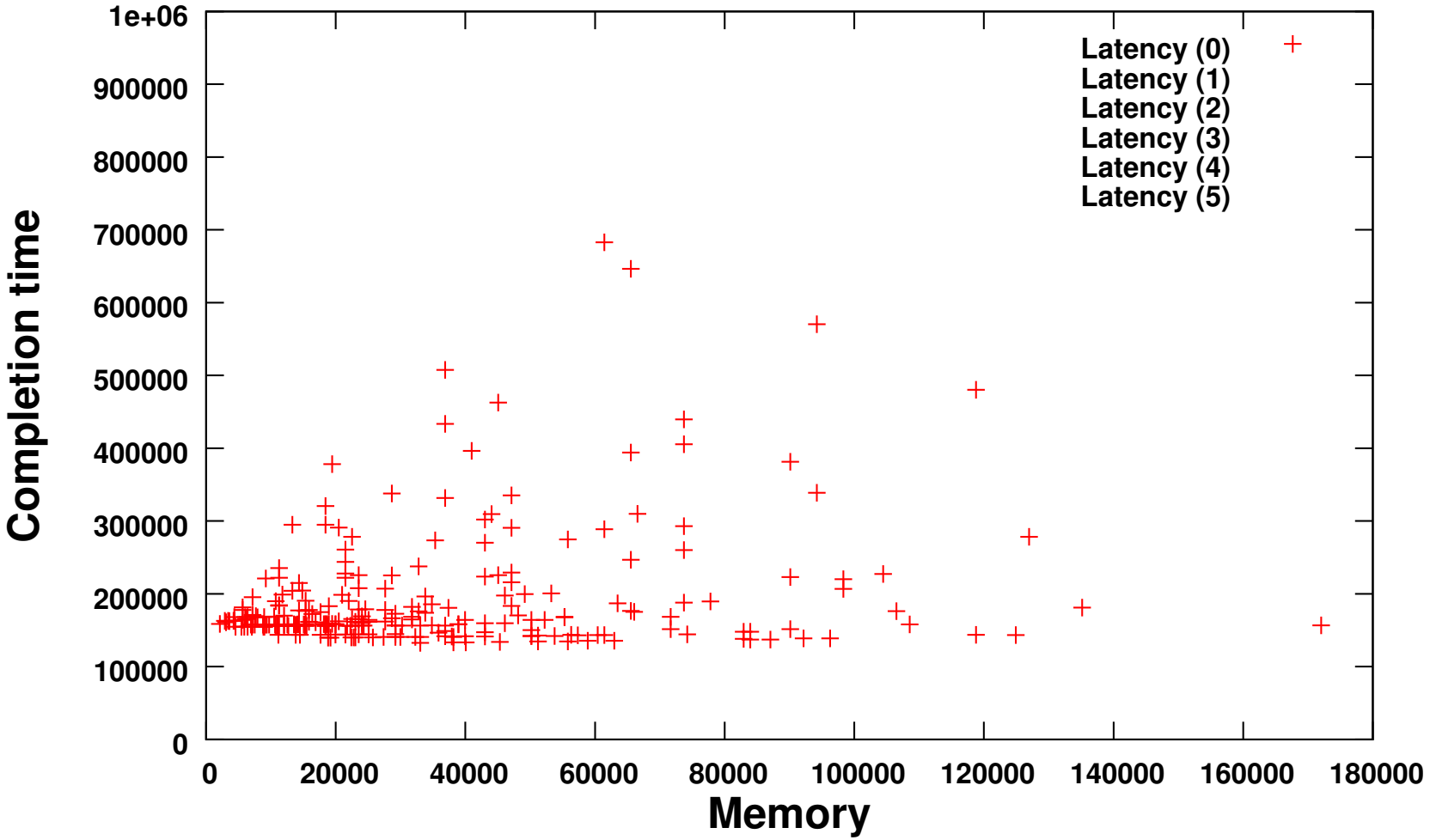
Future work :

- Improve the optimizations
- Manage parallelism
- Set up a framework to design network of communicating kernels

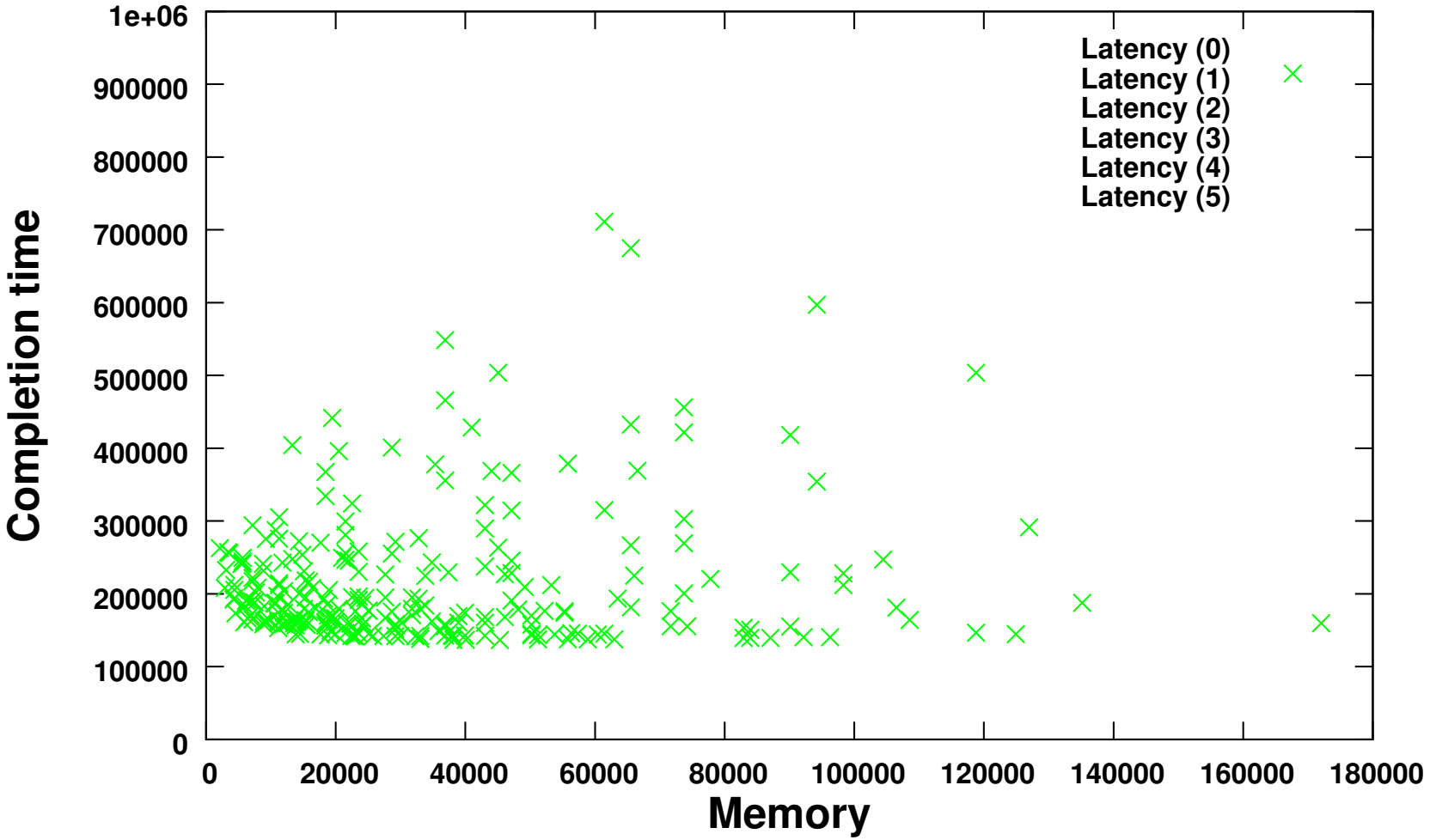
Optimization process details



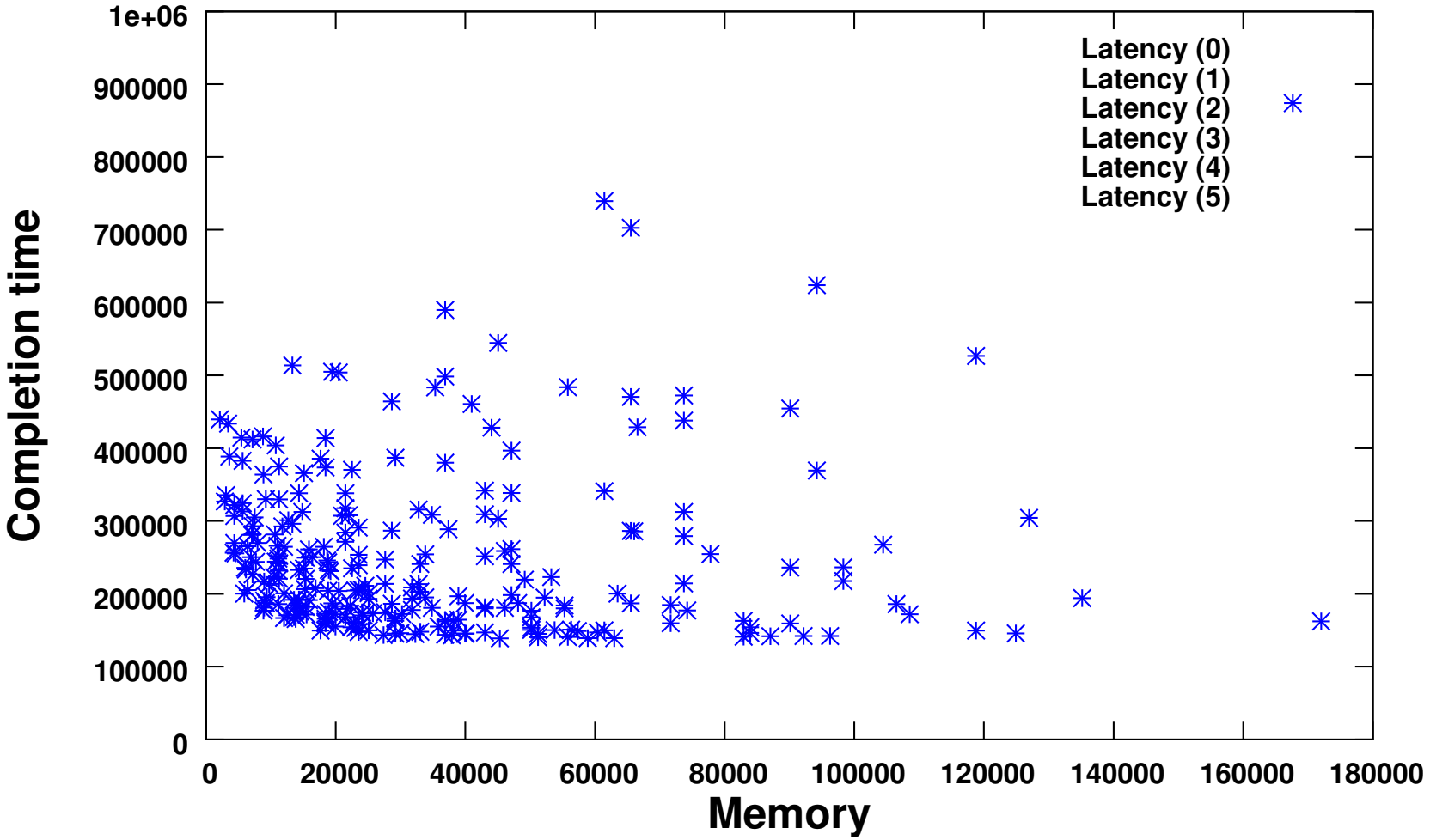
Choose a tiling ?



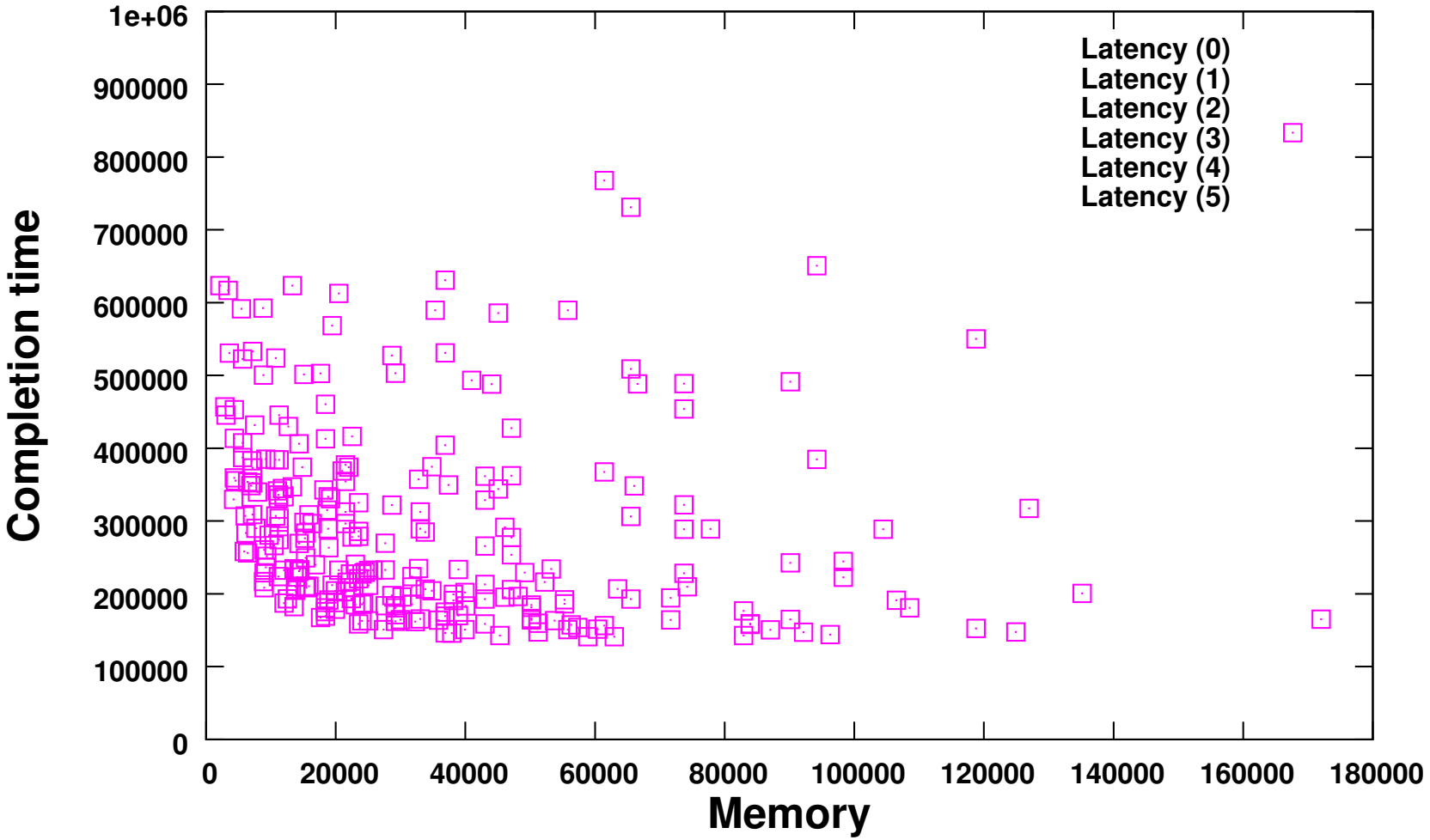
Choose a tiling ?



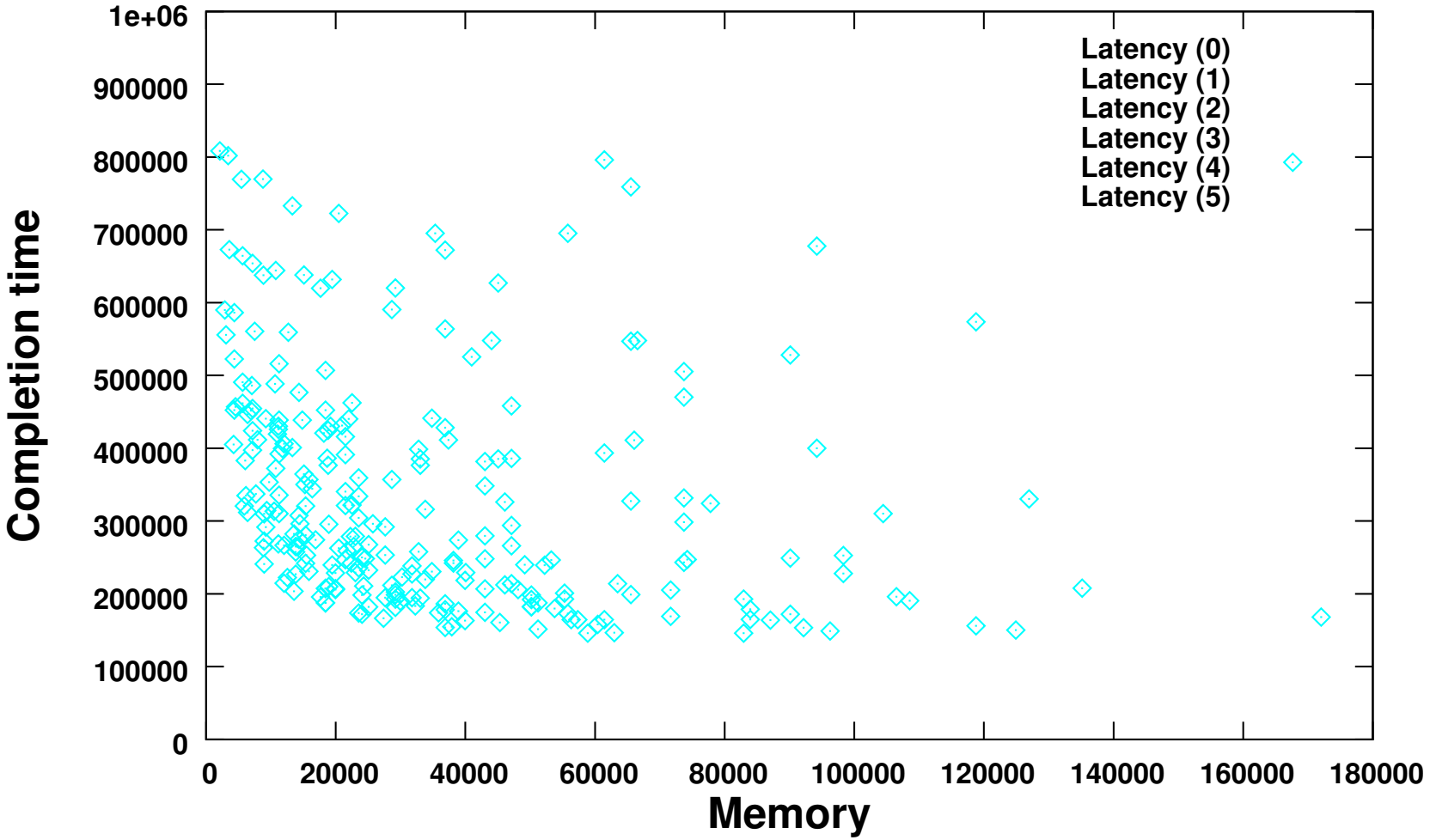
Choose a tiling ?



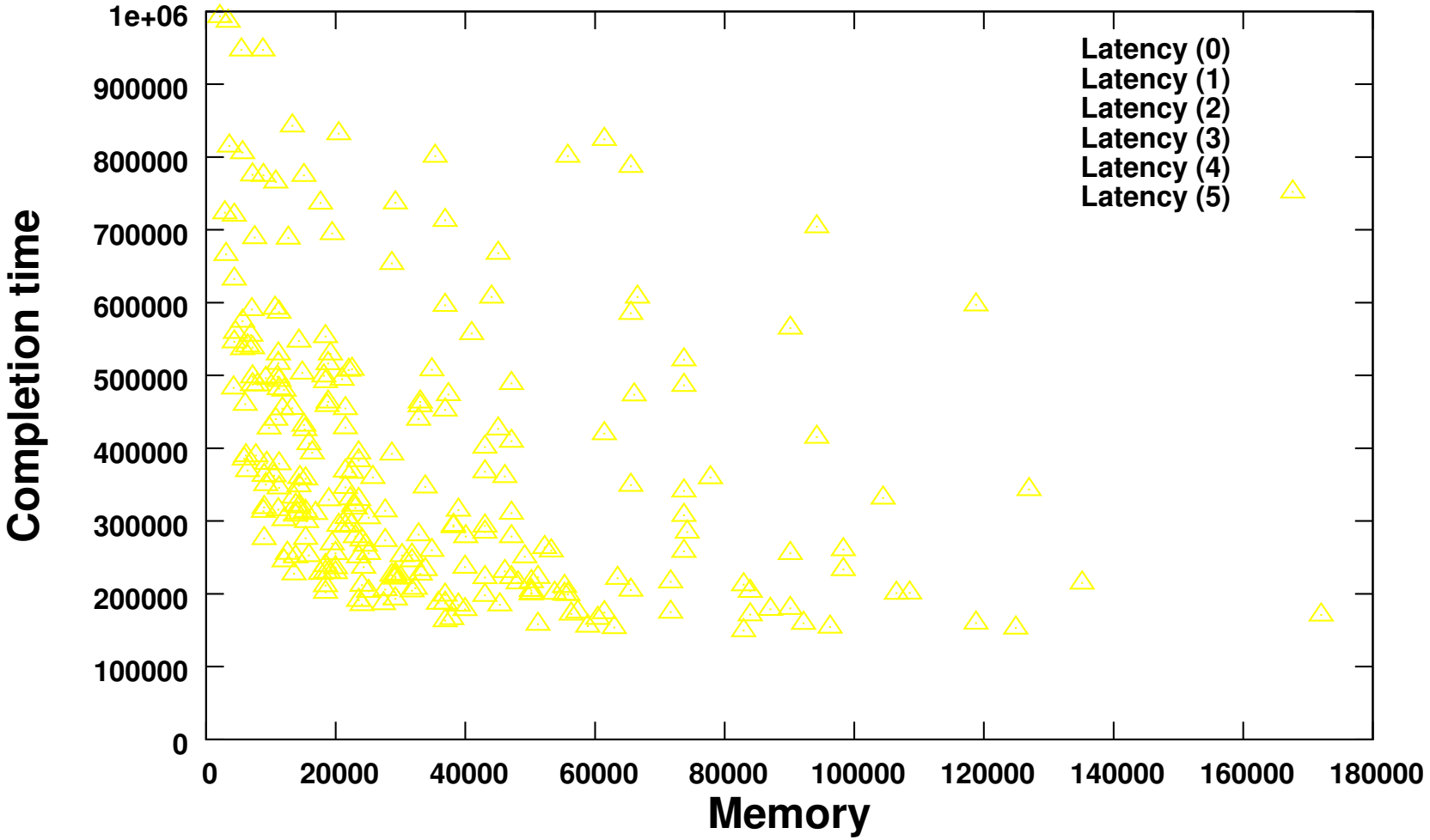
Choose a tiling ?



Choose a tiling ?



Choose a tiling ?



WASC 2012

Design of non-linear kernel IPs for vision systems

S. Mancini & F. Rousseau

Plan Détaillé

- Context
- Non-linear kernels
- Problems
- Problems II
- Goals
- The proposed method
- Target architecture
- Scheduling
- Optimizations
- Traffic minimisation
- Managing the disparity by breaking the regularity
- IT optimization
- Computing the IT size
- Trade-offs IT vs time
- Buffer optimization
- Trade-offs buffers vs time
- Benchmarks
- Mipmap ?
- Results
- Conclusion & perspectives
- Optimization process details
- Choose a tiling ?