Encadrant(s) CEA : Maha Kooli, CEA Grenoble
Directeur de thèse : Henri-Pierre Charles, CEA Grenoble
École doctorale : ED MSTII (Université Grenoble Alpes)
Laboratoire d'accueil : CEA/DRT/DSCIN/LFIM, CEA Grenoble
Date de soutenance : 10 Mars 2023

Rapporteurs :
Lionel Lacassagne, Sorbonne Université (Paris VI)
Alberto Bosio, Ecole Centrale Lyon

Examinateurs :
Caroline Collange, INRIA Rennes
Laure Gonnord, Grenoble INP (ENSISAR)
Frédéric Petrot, Université Grenoble Alpes (TIMA)

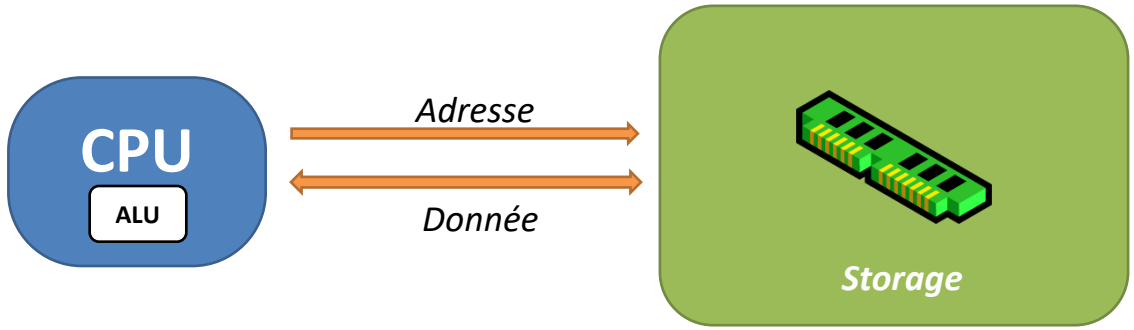
Directeur de Thèse :
Henri-Pierre Charles, CEA Grenoble

DE LA RECHERCHE À L'INDUSTRIE

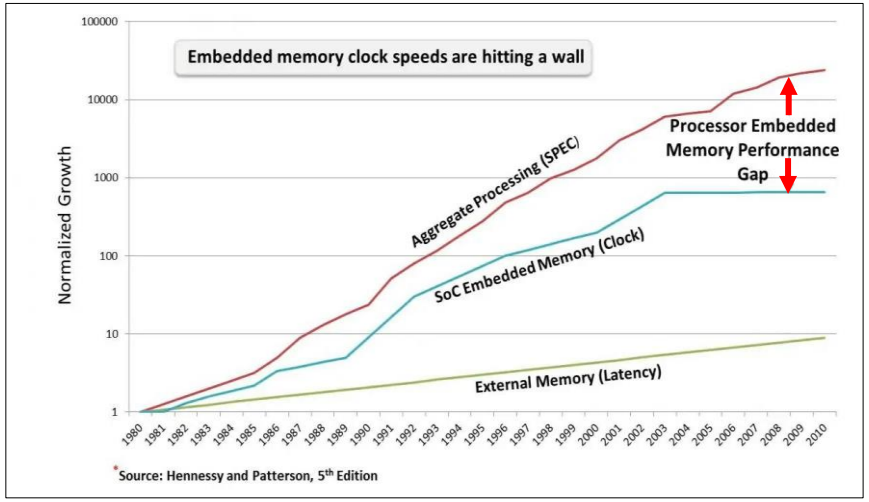
Modèle de programmation bas-niveau pour architectures de calcul proche-mémoire

Kévin Mambu

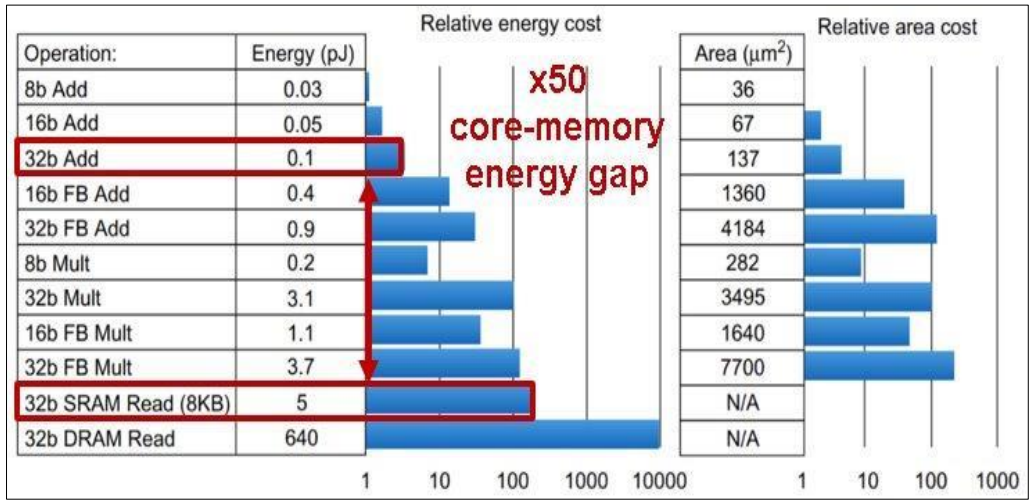
Motivations architecturales – Goulot d'étranglement de von Neumann



Une architecture de von Neumann



***Memory Wall** [2] : limitation de la performance d'exécution*

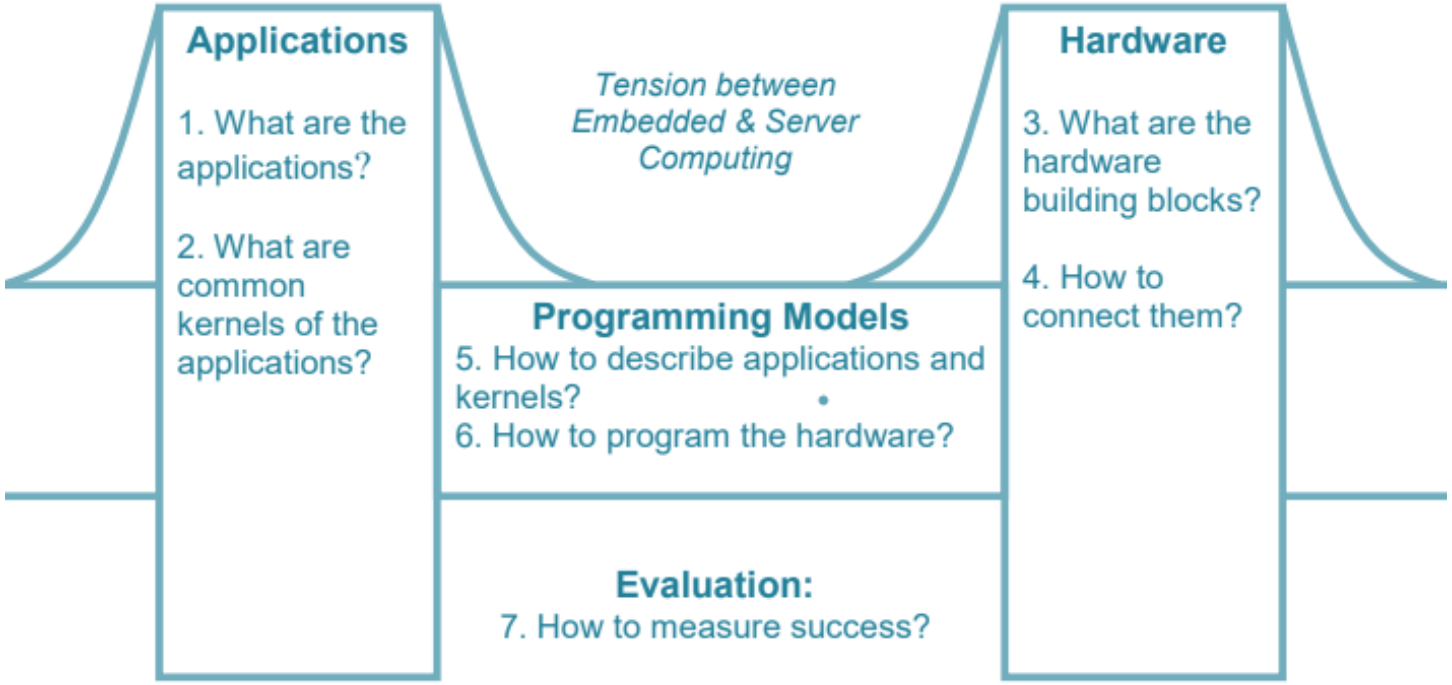


***Energy Wall** [3] : limitation de la performance énergétique*

[2] Hennessy & Patterson, Computer Architecture : A Quantitative Approach (4th edition). 2011.

[3] Mark Horowitz, Computing's Energy Problem (and What Can We Do About It), IEEE ISSCC 2014.

Motivations initiales de la thèse

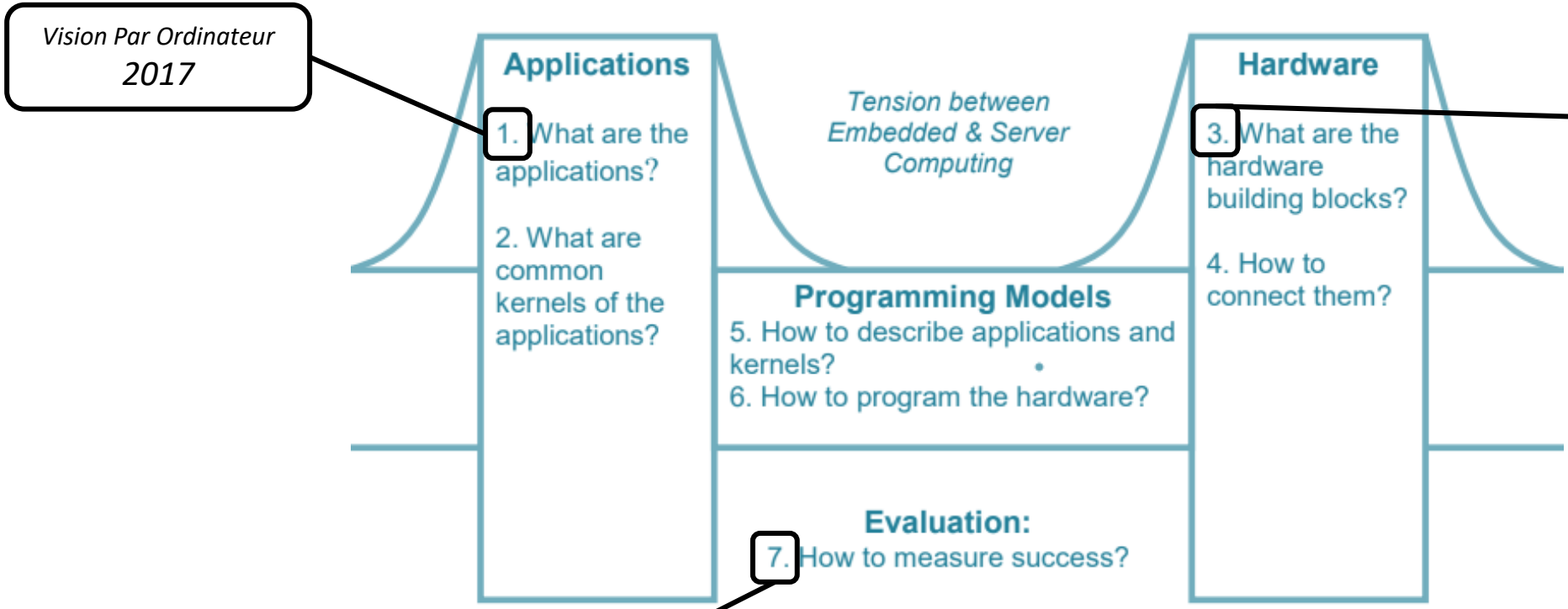


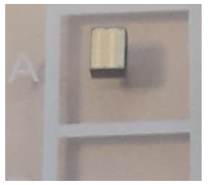
La tension entre logiciel et matériel supportée par les modèles de programmation [1].

[1] Asanovic et al, The Landscape of Parallel Computing Research : A View from Berkeley. 2006.

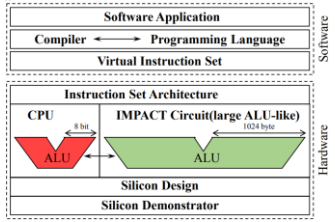
Motivations initiales de la thèse

Vision Par Ordinateur
2017





1er circuit-test [3]
2020

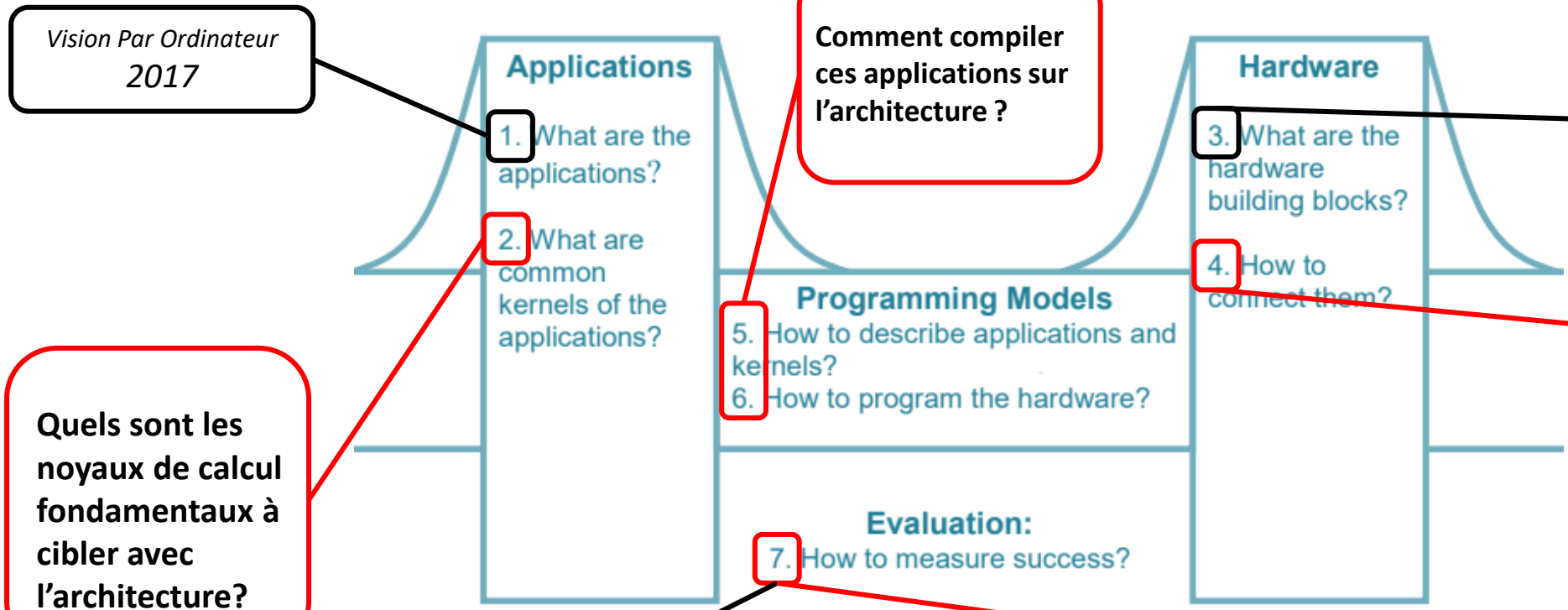


1er modèle de simulation [2]
2017

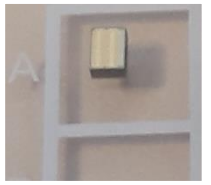
[2] Kooli et al., Software Platform Dedicated for In-Memory Computing Circuit Evaluation. RSP 2017.
[3] Noel et al., A 35.6TOPS/W/mm² 3-Stage Pipelined Computational SRAM with Adjustable Form Factor for Highly Data-Centric Applications. IEEE LSSC 2020.

Motivations initiales de la thèse

Vision Par Ordinateur
2017



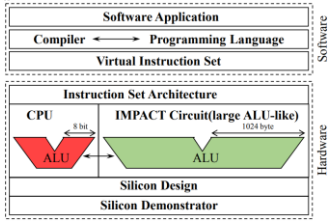
Comment compiler ces applications sur l'architecture ?



1er circuit-test
2020

Quels sont les noyaux de calcul fondamentaux à cibler avec l'architecture?

Comment interfacer l'architecture au sein d'un système complet ?



1er modèle de simulation
2017

Comment modéliser un système complet intégrant le calcul proche-mémoire?

Outline

Contributions :

- **Développement d'une plateforme de modélisation et de simulation basée sur QEMU**, pour l'évaluation de systèmes intégrant la mémoire C-SRAM.

- **Proposition d'une interface de programmation pour le Data-locality Management Unit (DMU),** un contrôleur mémoire spécialisé pour le transfert de données de stencils entre une mémoire DRAM et une mémoire C-SRAM**.**

- **Intégration et implémentation de passes de transformations au sein de Hybrogen**, un environnement de compilation dynamique pour apporter le support de la mémoire C-SRAM et du DMU.

- **Evaluation expérimentale** des contributions susmentionnées.

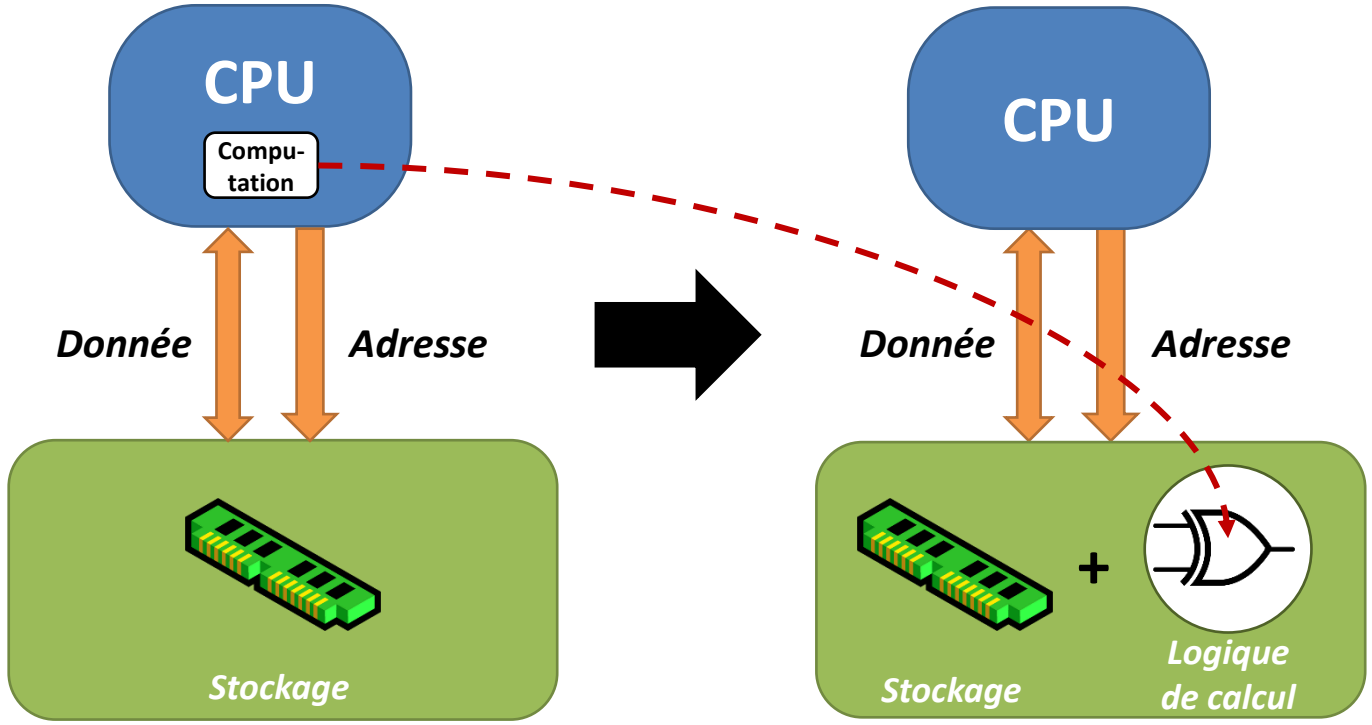
Sommaire

Une solution matérielle proposée – Le In-Memory Computing (IMC)

$Delai_{CPU}$

$>$

$Delai_{MEM}$



$Ops/cycle_{CPU}$

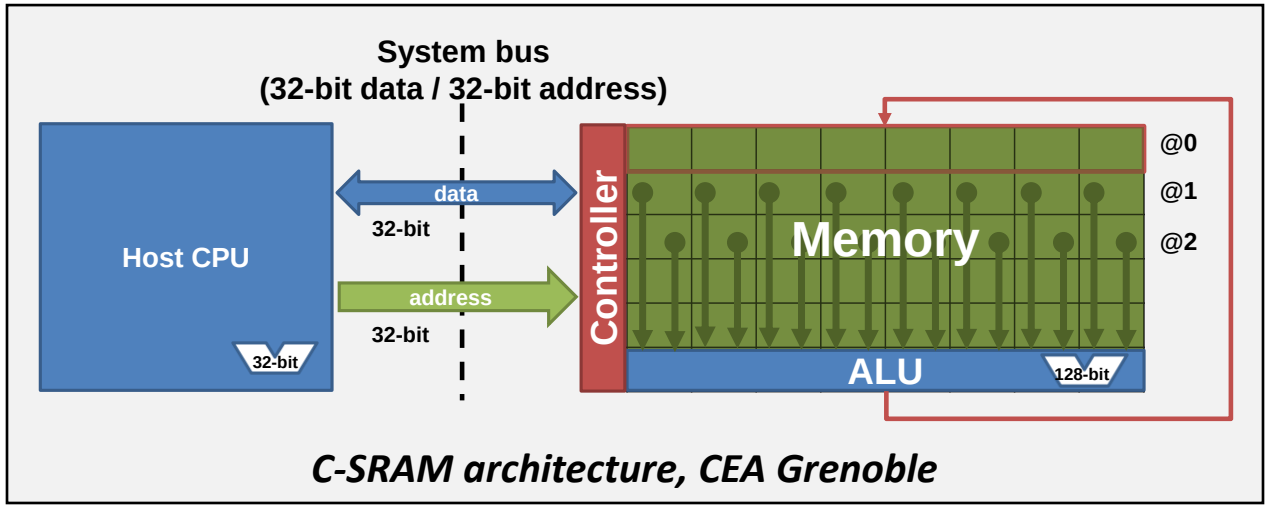
$\ll$

$Ops/cycle_{MEM}$

- Intégration du calcul au sein des mémoires
- Reduction de la consommation de bande passante
- Obtention d'un degré de parallélisme de calcul plus important que le CPU

[12] A 35.6 tops/w/mm² 3-stage Pipelined Computational SRAM with Adjustable Form Factor for Highly Data-Centric Applications. IEEE Solid-State Circuits Letters, 3 :286–289, 2020.

Architecture Computational SRAM (C-SRAM) [21][22][23][24]



[21] Kooli et al, Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution, ACM JETC 2022

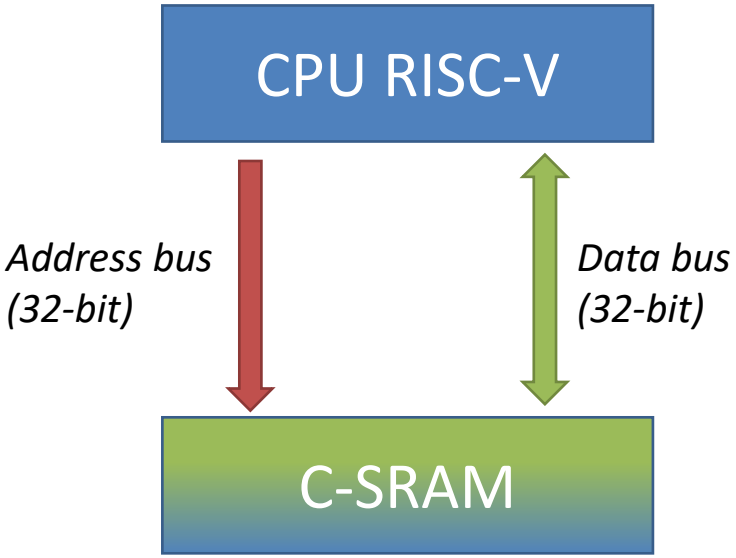
[22] Gauchi et al., Reconfigurable tiles of computing-in-memory SRAM architecture for scalable vectorization, ISLPED 2020.

[23] Noel et al., A 35.6 TOPS/W/mm² 3-Stage Pipelined Computational SRAM With Adjustable Form Factor for Highly Data-Centric Applications, ISSCC 2020

[24] Egloff et al., Storage Class Memory with Computing Row Buffer: A Design Space Exploration, DATE 2020.

- *Logique de calcul en cellules SRAM sur-mesure / périphérie SRAM*
- *Supporte les opérations logiques, de 1 à 128 bits, et arithmétiques vectorielles, de 8 à 32 bits.*
- *Programmée par le CPU grâce au bus mémoire → peut être couplée a n'importe quelle architecture CPU*

Architecture Computational SRAM (C-SRAM)



Exemple ci-contre :
$freq_{C\text{-}SRAM}$ = X, $freq_{CPU}$ = 6X.

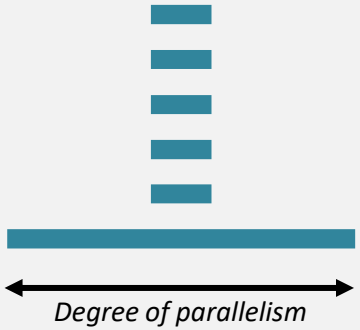
Bus adresse (32-bit) *Bus donnée (32-bit)*

64	58	50	34	32	16	0
100000	OPERANDE	RANGEE DESTINATION	00	IMMEDIAT 32-BIT		
100000	OPERANDE	RANGEE DESTINATION	00	IMMEDIAT 16-BIT	SOURCE 1	
100000	OPERANDE	RANGEE DESTINATION	00	SOURCE 2	SOURCE 1	

Flot d'exécution RISC-V

```
lui r0, #CM_MUL8_OPC
slli r2, #ROW0, 2
or r0, r0, r2
slli r1, #ROW1, #16
or r1, r1, #ROW2
sw r1, 0(r0)
```

Débit de calcul

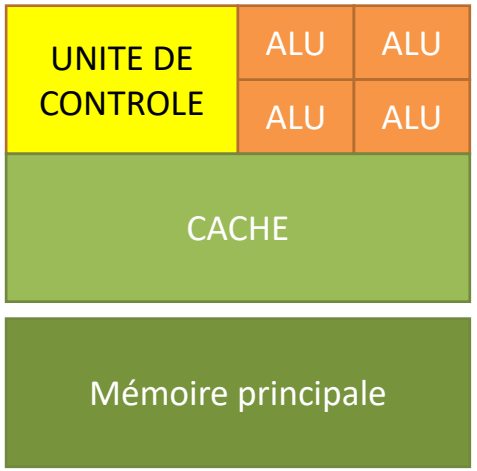


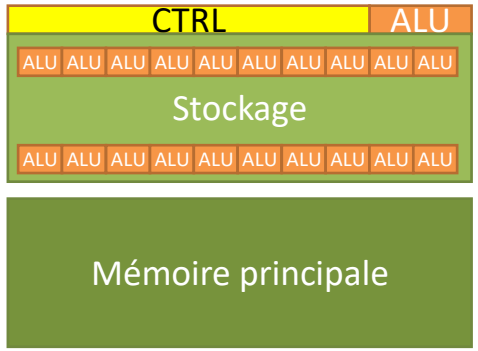
Unité C-SRAM

```
mul8 ROW0, ROW1, ROW2
```

Generation of C-SRAM instructions using dynamic addresses

In-Memory Computing (IMC) – Comparaison avec des solutions existantes



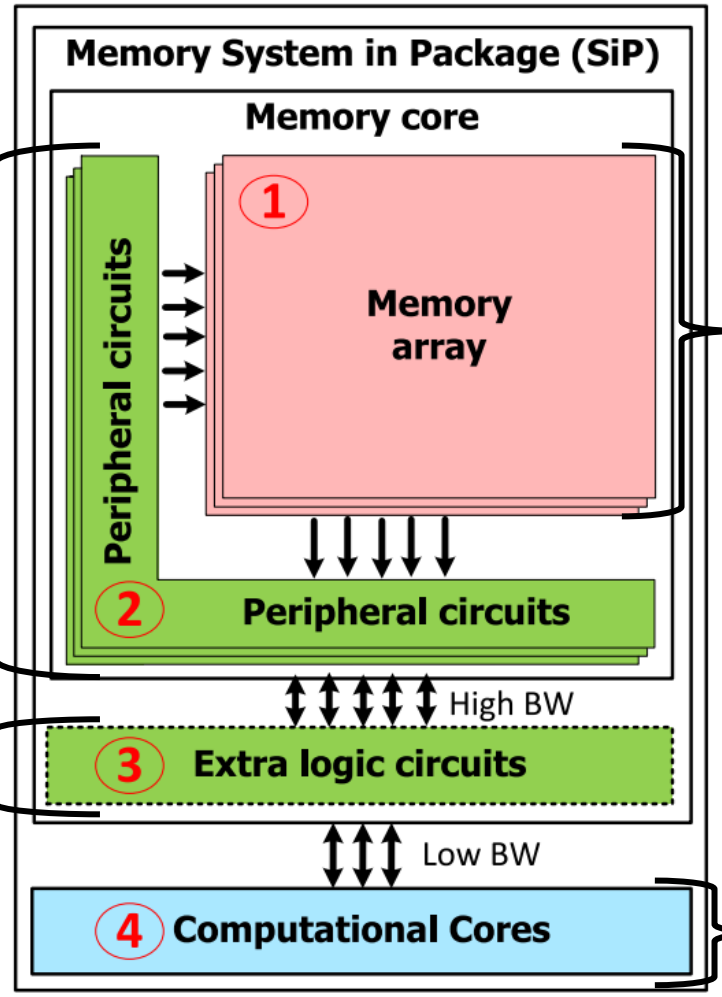


	Système avec processeur graphique (GPU)	*Système avec processeur généraliste (CPU)*	*Système avec mémoire IMC*
Architectures cibles	HPC	HPC / embarqué	Faible consommation
Degré de parallélisme	Très large / Très découplé (SIMT non-bloquant)	Assez large (SIMD / MIMD)	Extrêmement large / Très couplé (SIMD / systolique)
Arithmétique numérique de prédilection	Virgule flottante	Virgule flottante / entier (64-bit jusqu'à 8-bit)	Entier (32-bit jusqu'a 1-bit)
Organisation architecturale	Macro (niveau cœurs)	Milli (niveau composants)	Micro (niveau mémoire)
Modèle de programmation	CUDA	Langages généralistes + APIs threads	?

In-Memory Computing (IMC) – Taxonomie des architectures IMC [13]

Type-2 IMC
- Logique de calcul == logique périphérique (ex : décodage)
- Calcul numérique
- Non-altération des cellules mémoire

Type-3 IMC (Near-Memory Computing [NMC])
- Logique de calcul en connexion directe avec la logique périphérique
- Procédures numériques
- Bus à connexion 2D (rare), ou à empilement 3D



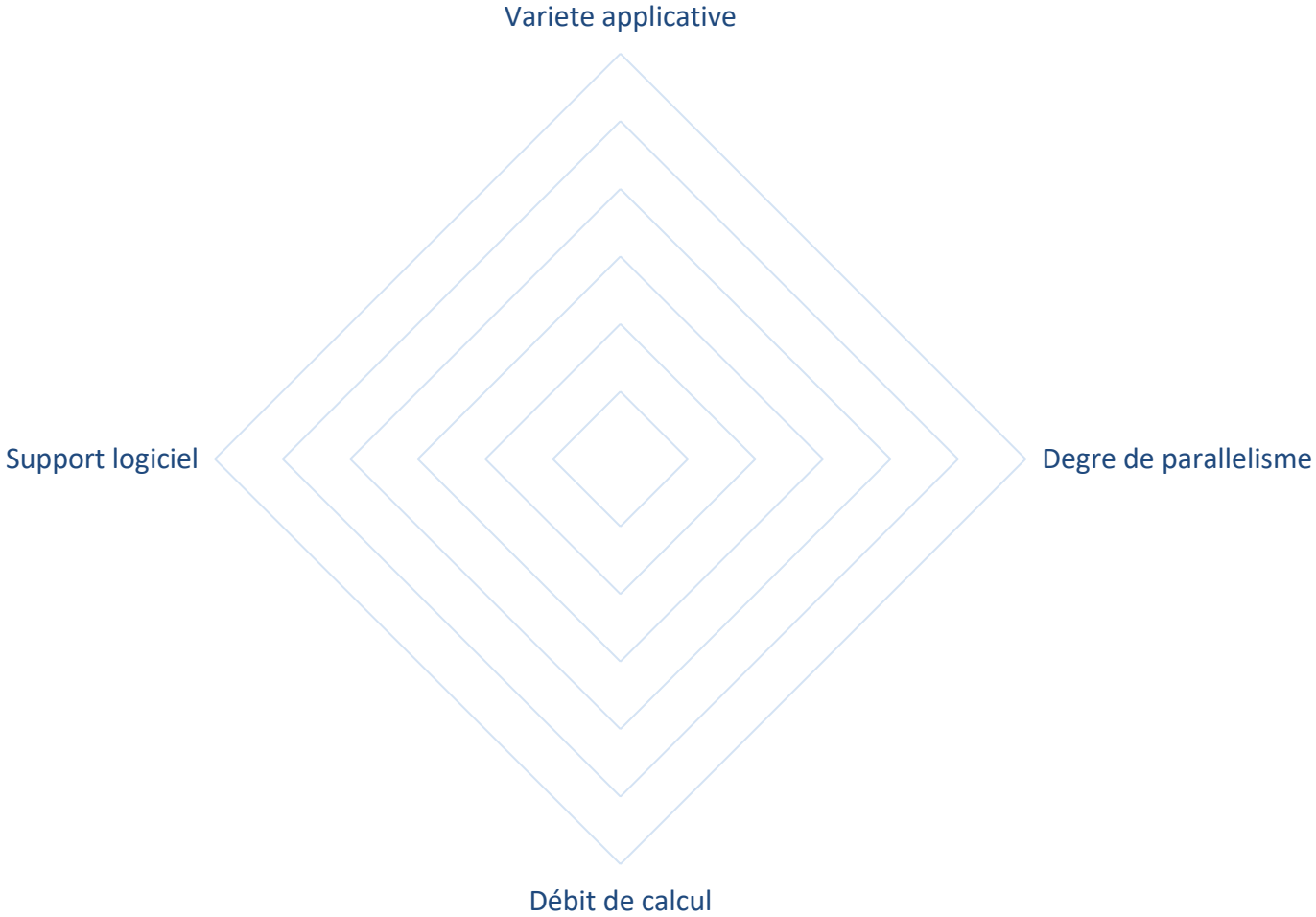
Type-1 IMC
- Logique de calcul == logique de stockage
- Calcul booléen
- Requiert des cellules mémoires sur-mesure

Architectures de von Neumann

[13] Gebregiorgis et al. – A Survey on Memory-centric Computer Architectures, ACM Journal on Emerging Technologies in Computer Systems (JETC) 10/2022

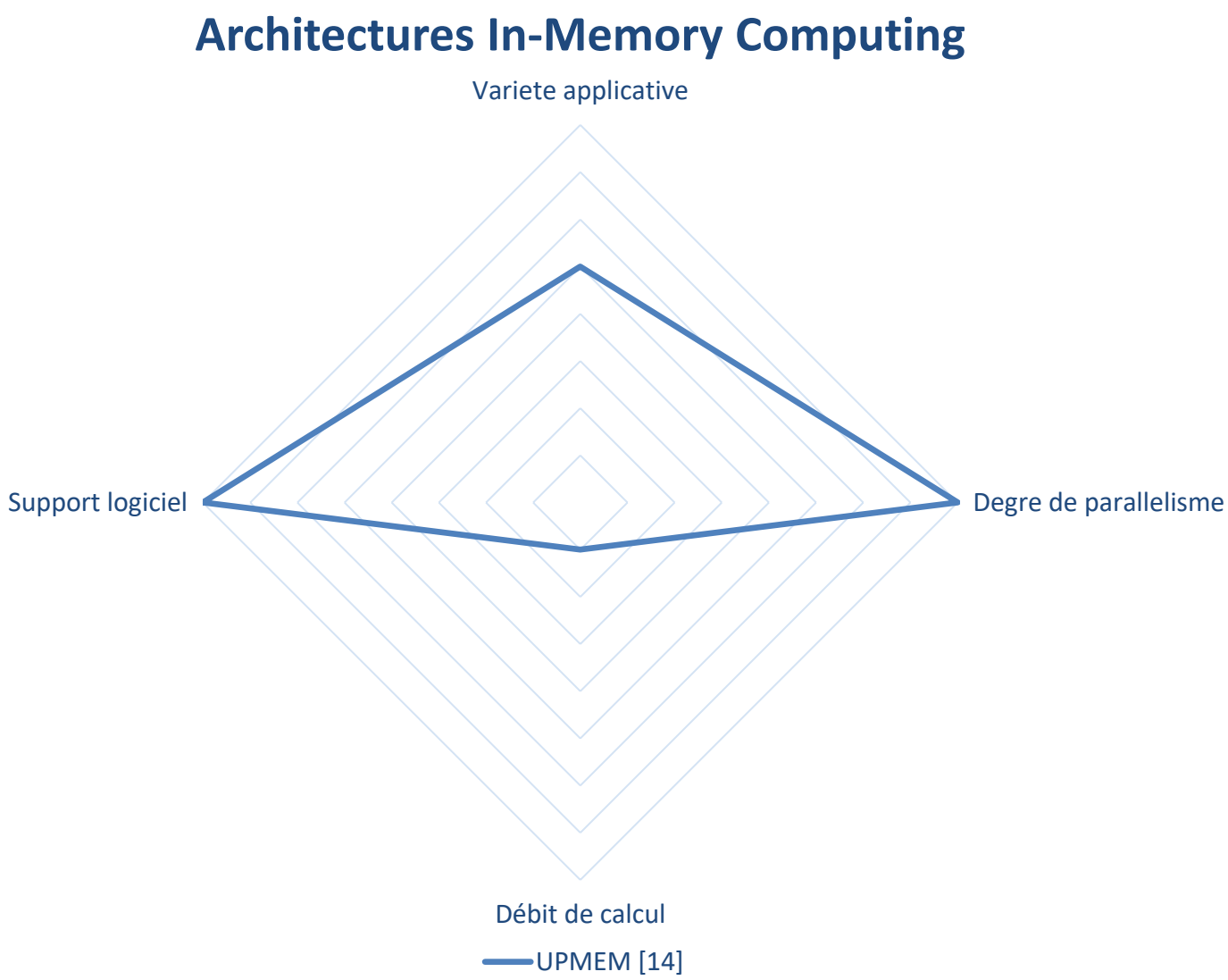
In-Memory Computing (IMC) – Etat de l'Art

Architectures In-Memory Computing



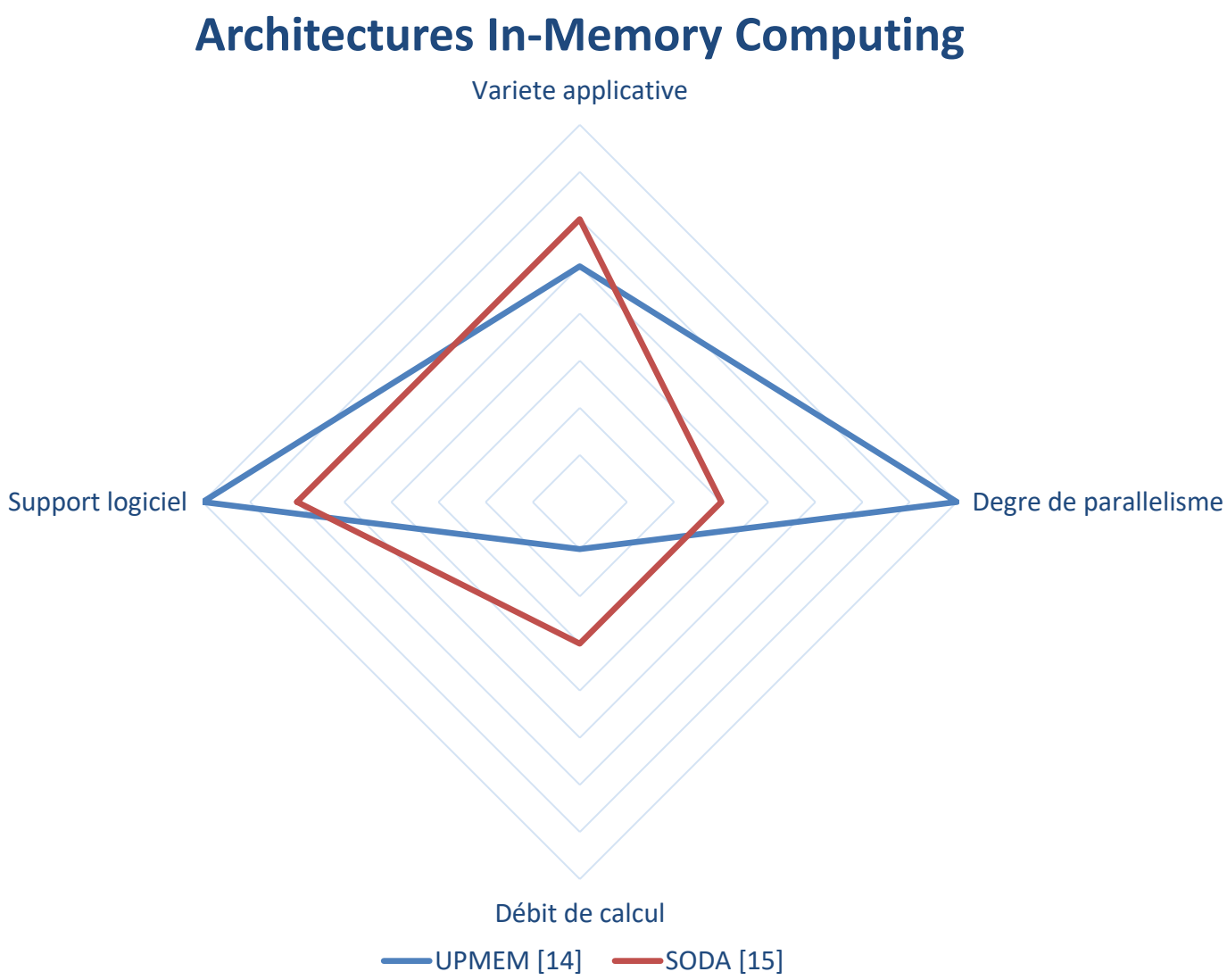
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.



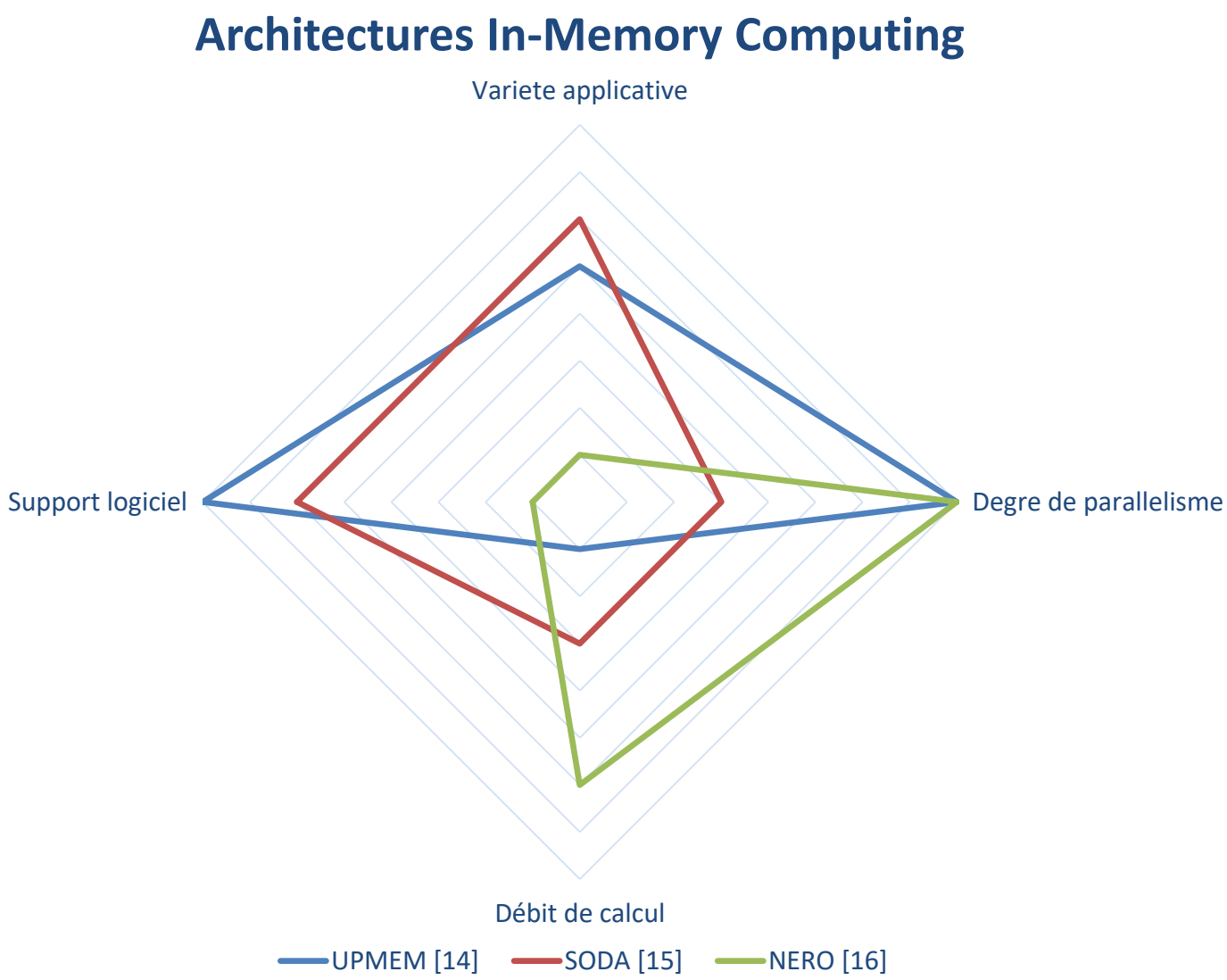
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.



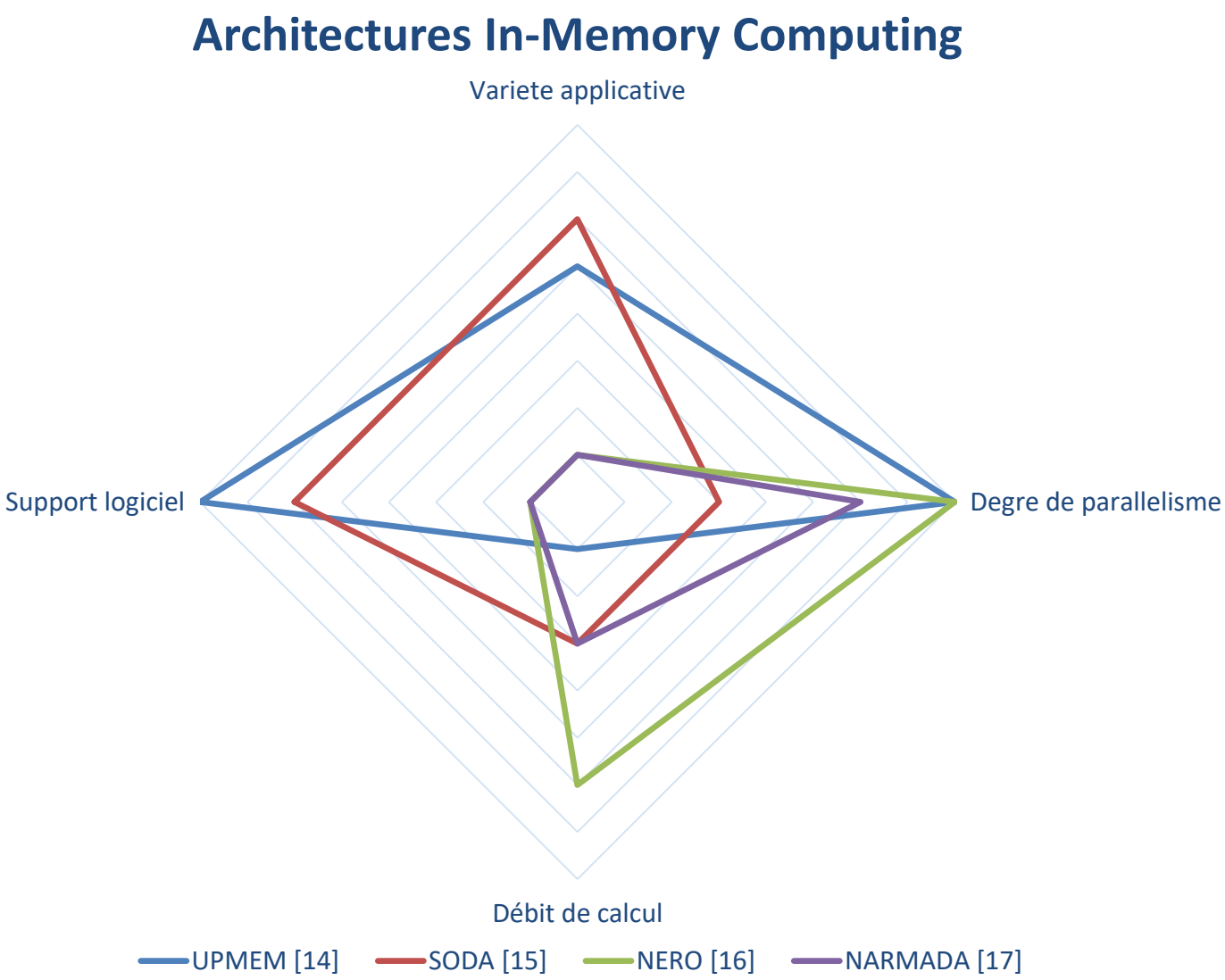
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.
[16] Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling. IEEE FPL 2020.



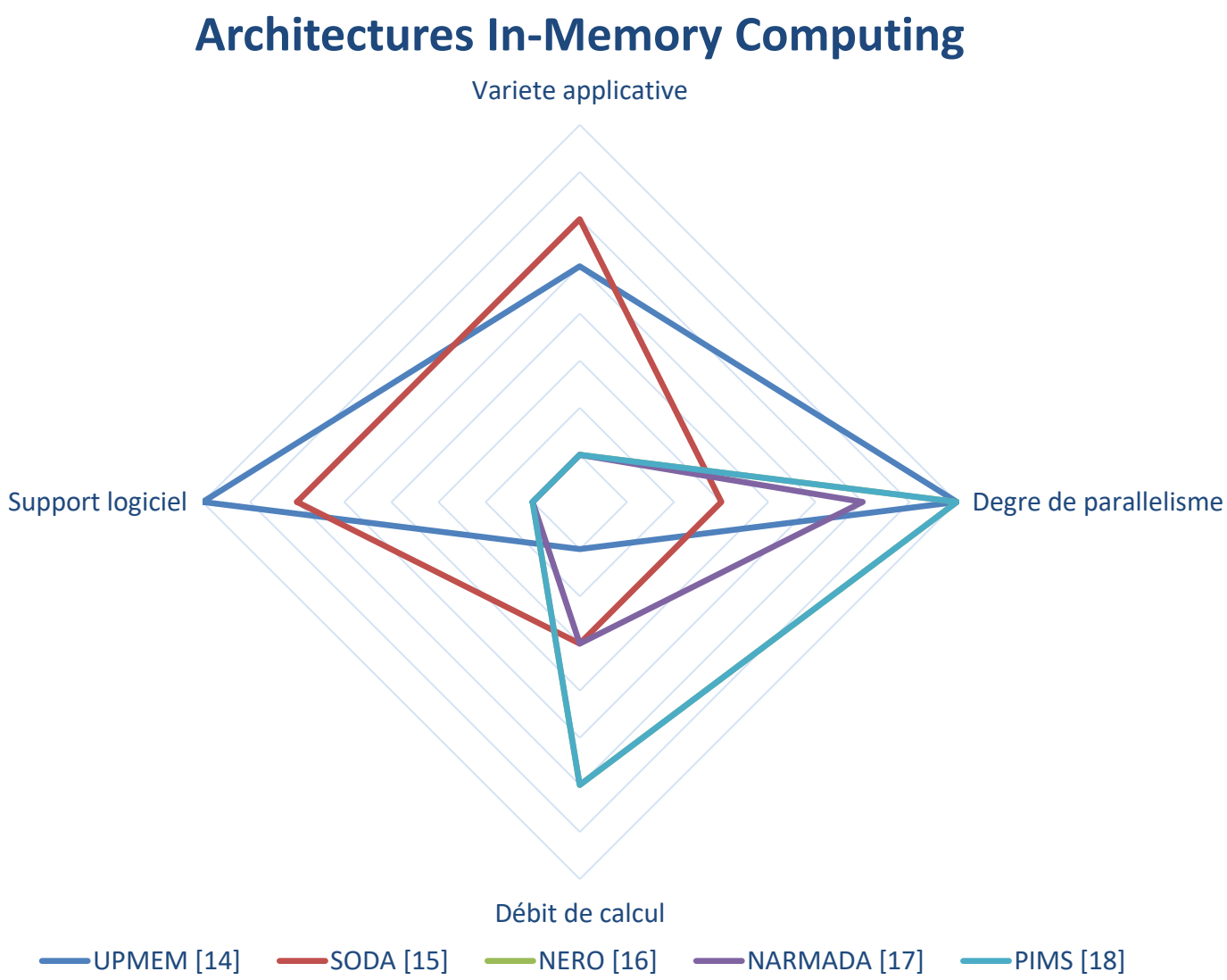
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.
[16] Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling. IEEE FPL 2020.
[17] Singh et al., NARMADA: Near-memory horizontal diffusion accelerator for scalable stencil computations. IEEE FPL 2019.



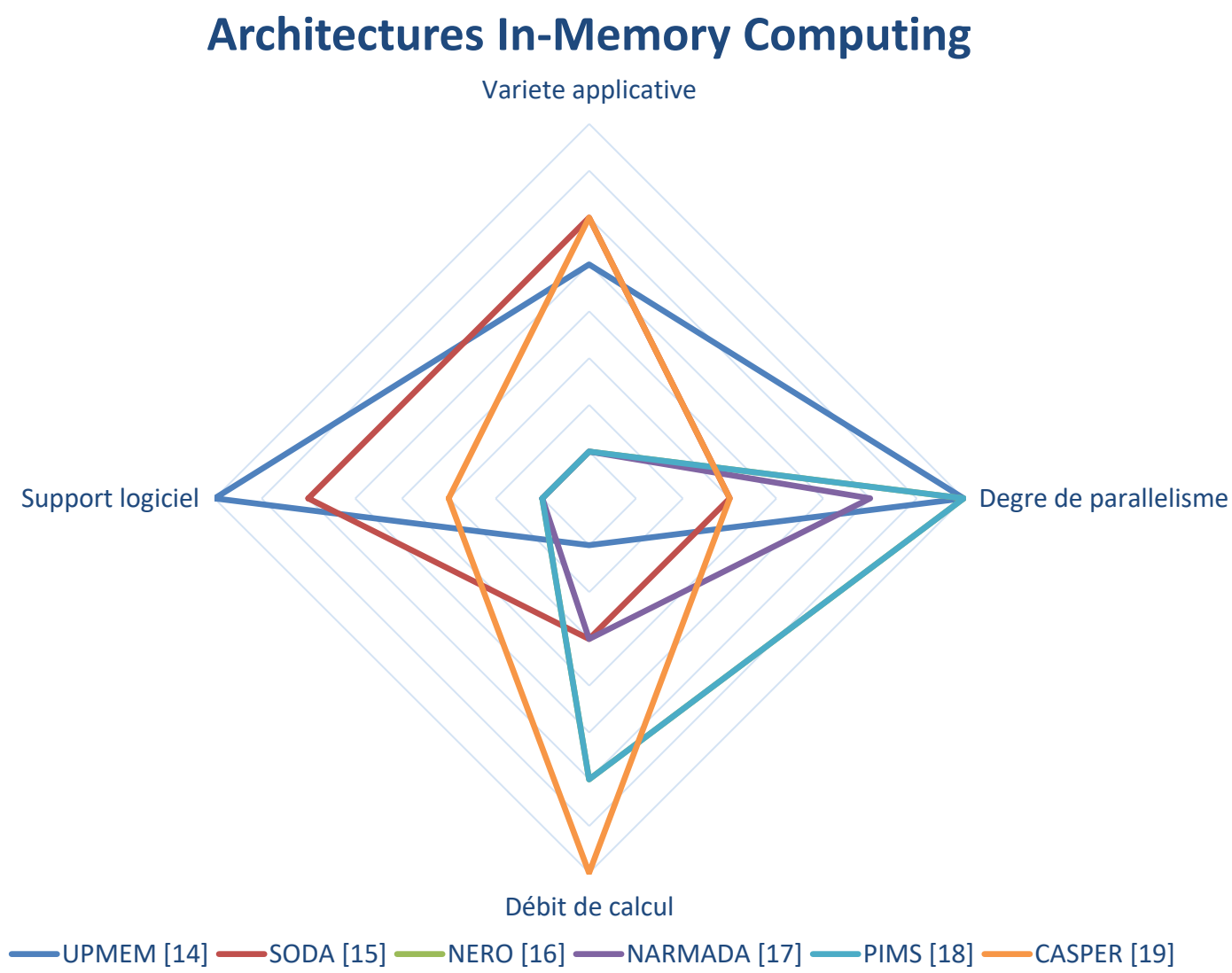
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.
[16] Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling. IEEE FPL 2020.
[17] Singh et al., NARMADA: Near-memory horizontal diffusion accelerator for scalable stencil computations. IEEE FPL 2019.
[18] Li et al., PIMS: a lightweight processing-in-memory accelerator for stencil computations. ACM MEMSYS 2019.



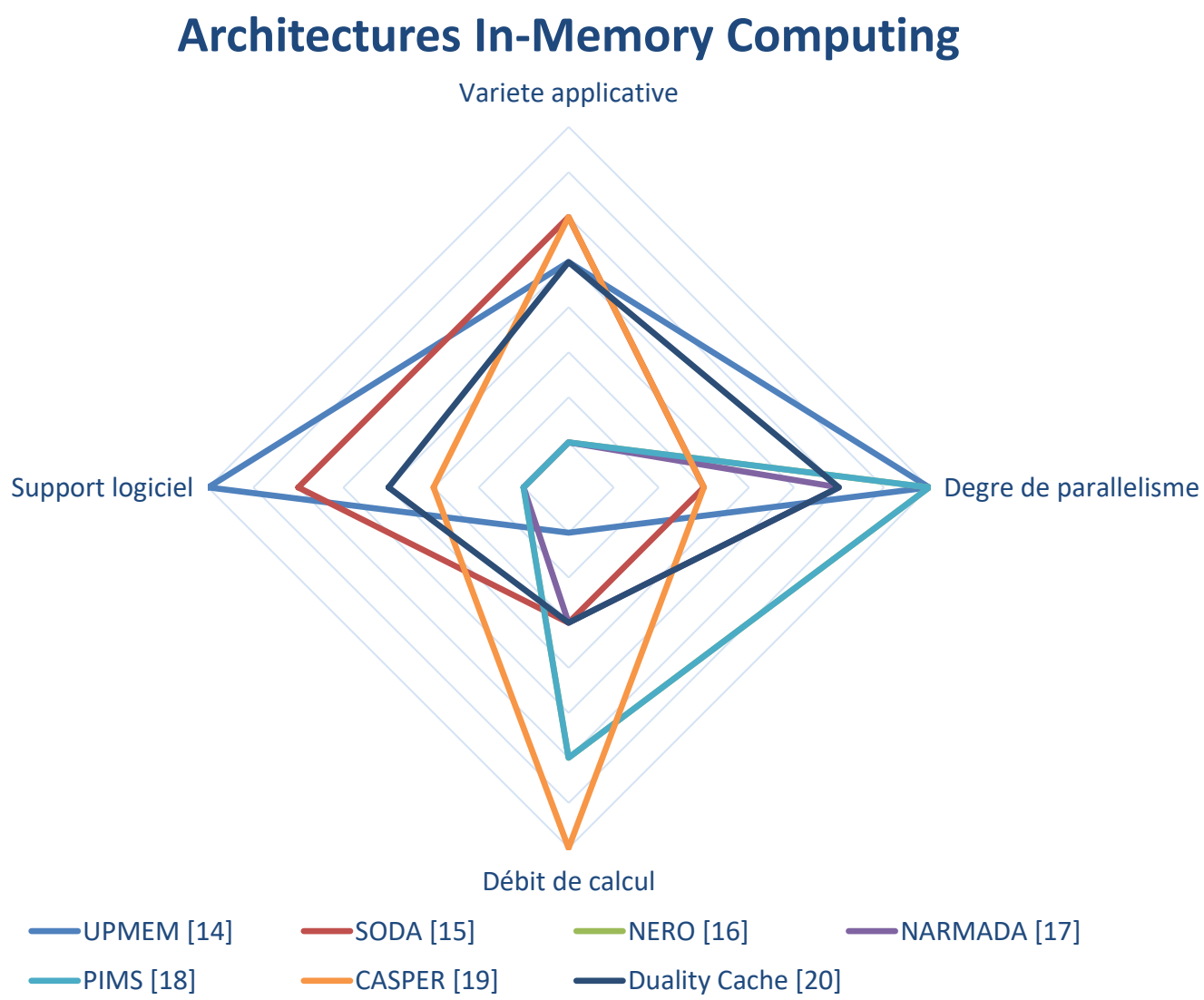
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.
[16] Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling. IEEE FPL 2020.
[17] Singh et al., NARMADA: Near-memory horizontal diffusion accelerator for scalable stencil computations. IEEE FPL 2019.
[18] Li et al., PIMS: a lightweight processing-in-memory accelerator for stencil computations. ACM MEMSYS 2019.
[19] Denzler et al., Casper: Accelerating Stencil Computation using Near-cache Processing. 2021.



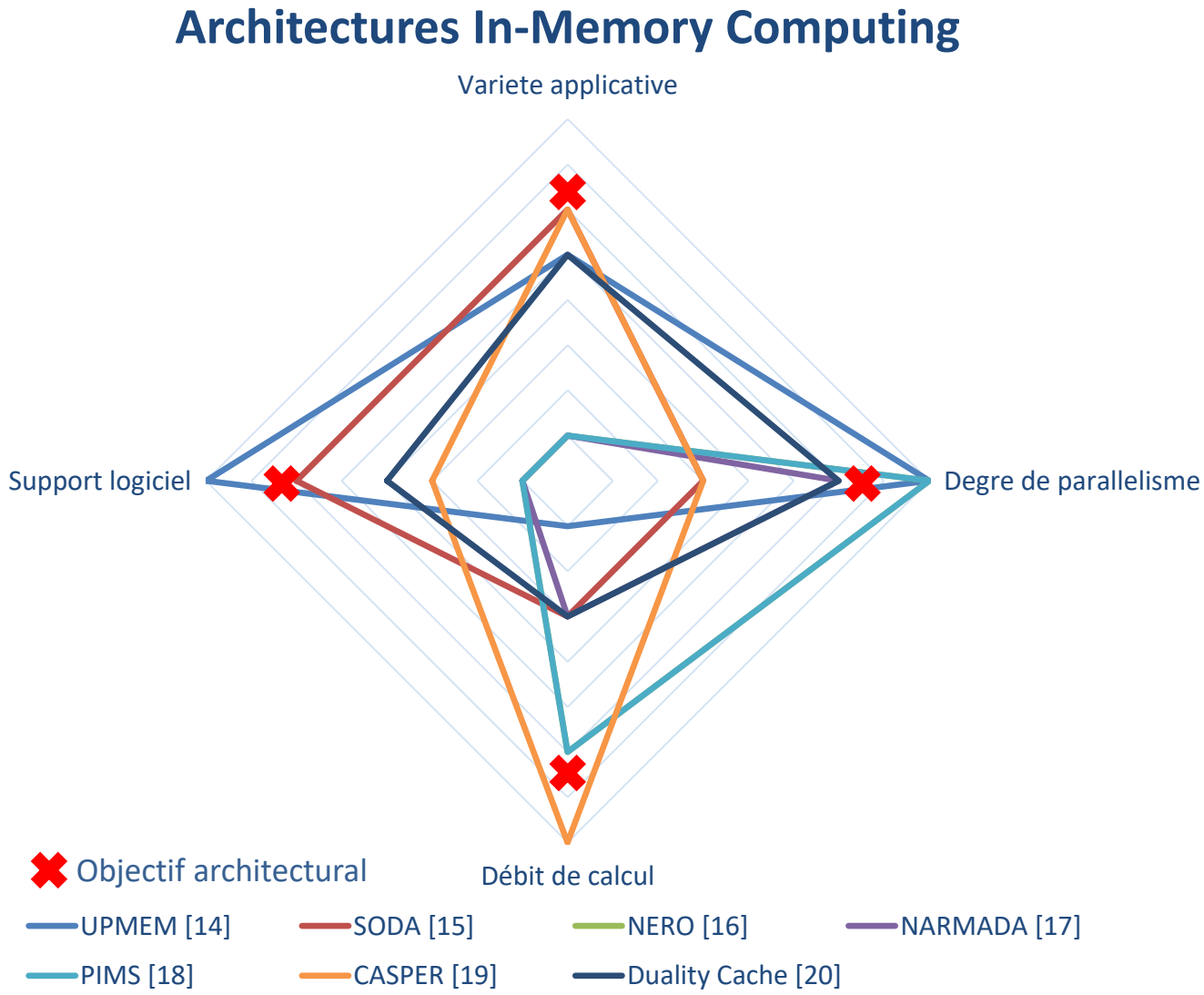
In-Memory Computing (IMC) – Etat de l'Art

[14] Gómez-Luna et al., Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. 2022.
[15] Chi et al., SODA: Stencil with optimized dataflow architecture. ACM ICCAD 2018.
[16] Singh et al., NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling. IEEE FPL 2020.
[17] Singh et al., NARMADA: Near-memory horizontal diffusion accelerator for scalable stencil computations. IEEE FPL 2019.
[18] Li et al., PIMS: a lightweight processing-in-memory accelerator for stencil computations. ACM MEMSYS 2019.
[19] Denzler et al., Casper: Accelerating Stencil Computation using Near-cache Processing. 2021.
[20] Fujiki et al., Duality Cache for Data-Parallel Acceleration. ACM ISCA 2019.



In-Memory Computing (IMC) – Etat de l'Art

- Solutions matérielles pour l'accélération de Vision par Ordinateur ➔ recours à l'IMC
- **Pas de solution matérielle IMC pour l'accélération généraliste de la Vision par Ordinateur** (c.-à-d. des codes arbitraires par leurs accès de données ou leurs opérations arithmétiques).
- Pas de solutions logicielle pour faciliter la compilation de telles applications sur architectures IMC.



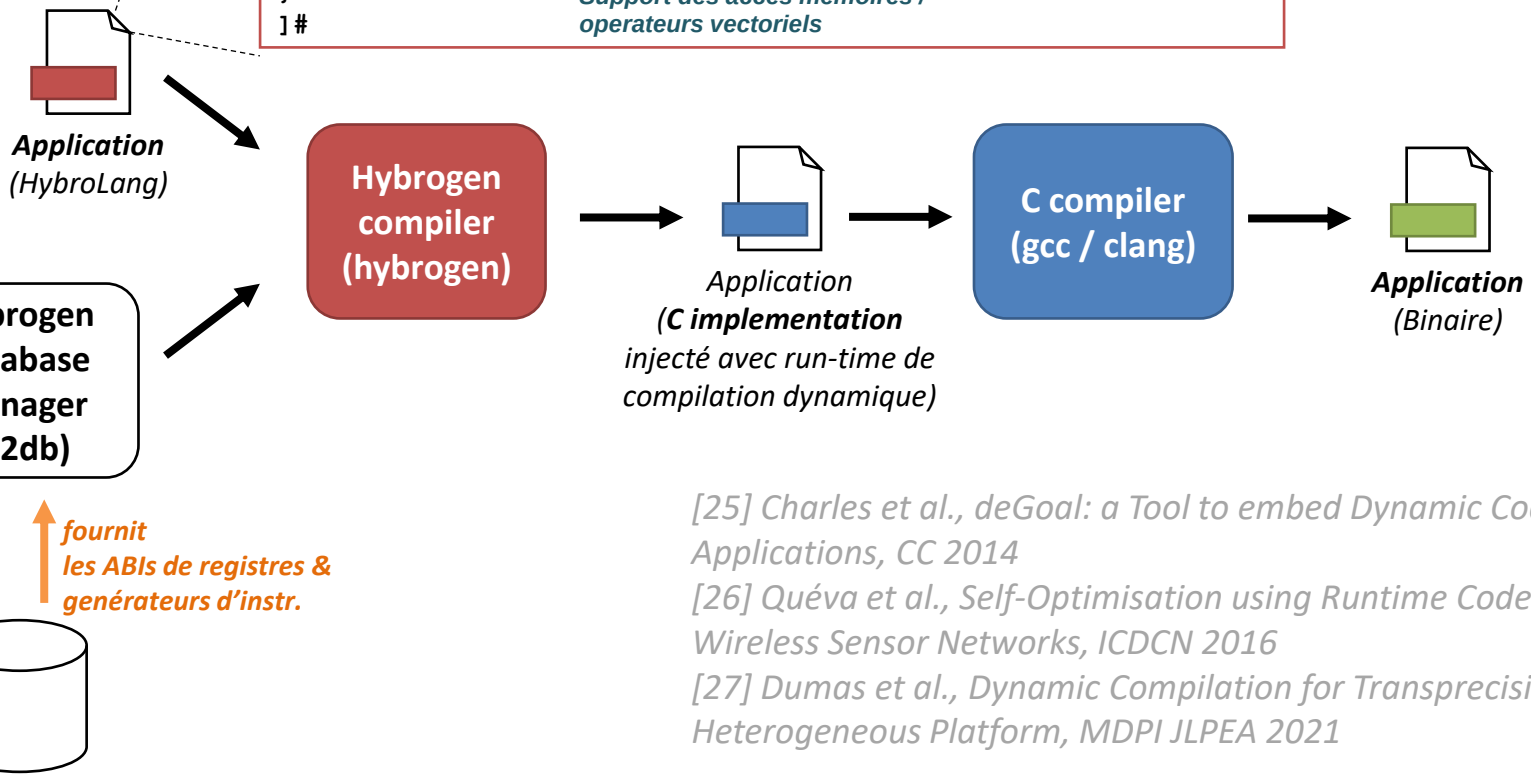
Hybrogen : Environnement de compilation dynamique [25][26][27]

- **Langage dédié pour la description de codes dynamiquement compilés**

```
#[
int 32 1 ImageDiff(sint[] 8 16 a, sint[] 8 16 b, sint[] 8 16
res, int 31 1 len)
{
        int 32 1 i;
        for(i = 0; i < len; i = i + 1)
        {
                res[i] = a[i] - b[i];
        }
        return 0;
}
]#
```

Taille de donnée — *Cardinalité* — *Types de données spécialisés*

Support des accès mémoires / operateurs vectoriels



Descriptions d'architectures :

- RISC-V
- Kalray MPPA
- IBM Power9
- etc...

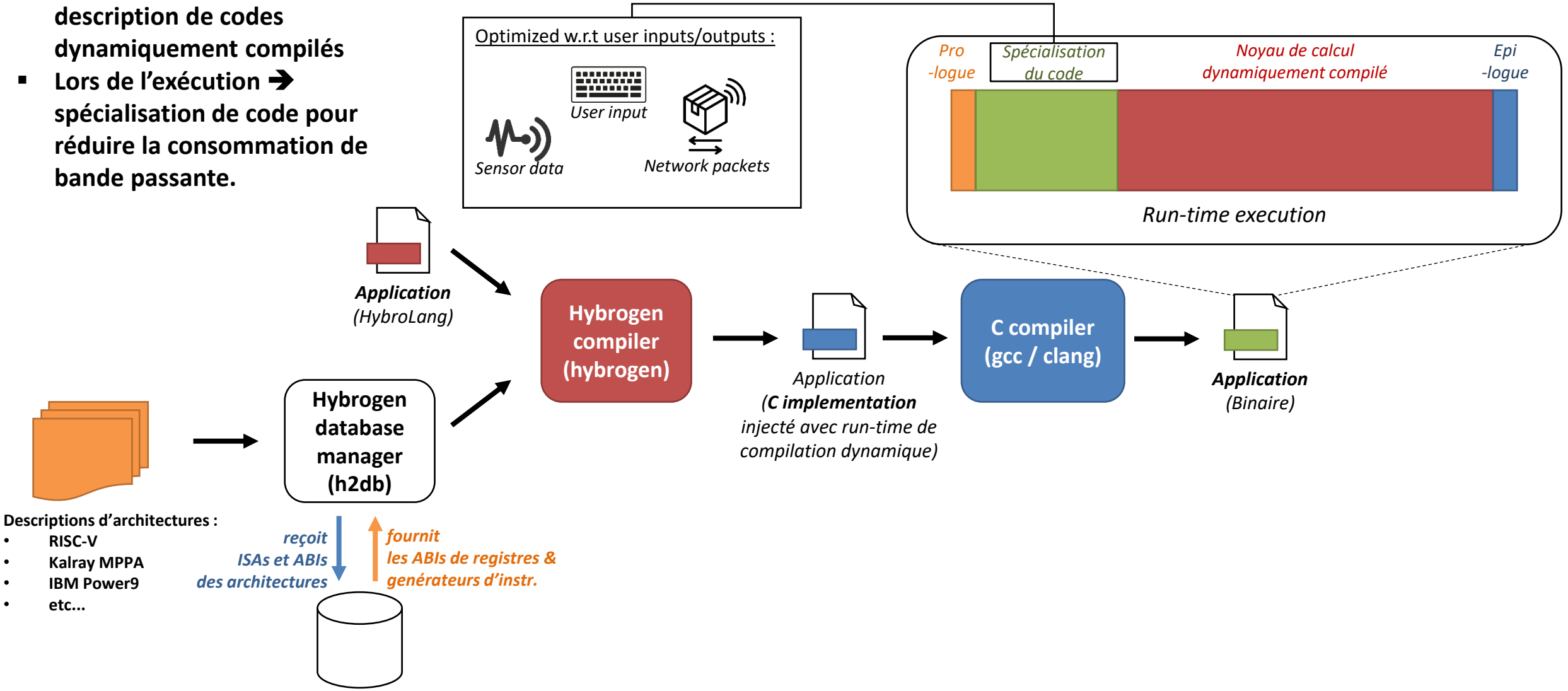
[25] Charles et al., deGoal: a Tool to embed Dynamic Code Generators into Applications, CC 2014

[26] Quéva et al., Self-Optimisation using Runtime Code Generation for Wireless Sensor Networks, ICDCN 2016

[27] Dumas et al., Dynamic Compilation for Transprecision Applications on Heterogeneous Platform, MDPI JLPEA 2021

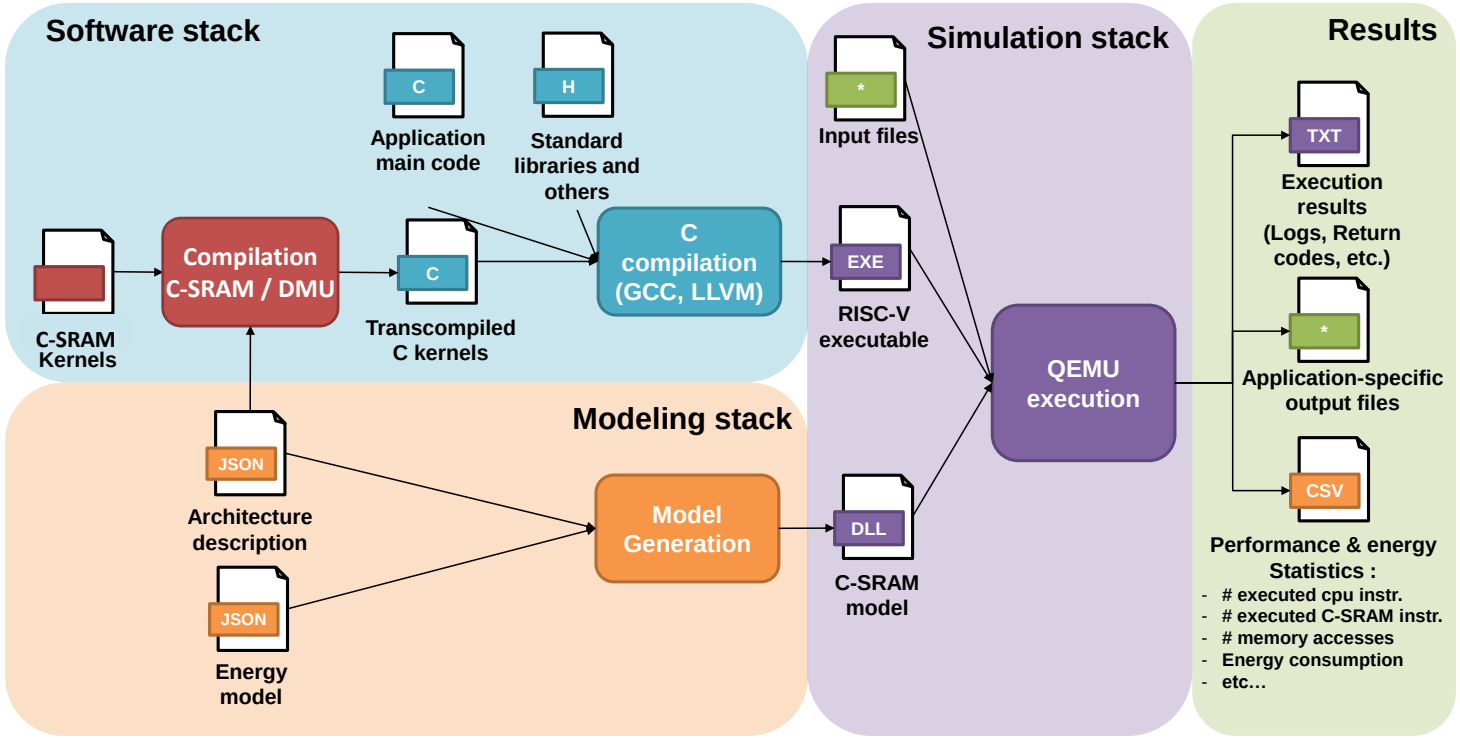
Hybrogen : Environnement de compilation dynamique

- **Langage dédié pour la description de codes dynamiquement compilés**
- **Lors de l'exécution ➔ spécialisation de code pour réduire la consommation de bande passante.**



Contribution – Plateforme de compilation et de simulation

- **Mise en place d'un méthodologie de simulation de jeux d'instructions, basée sur QEMU [32].**
- **Simulation de la C-SRAM générée à partir de descriptions d'architectures avec mémoires cache.**
- **Modèle de performance basée sur des données de l'Etat de l'art, datasheets, caractérisations physiques, etc.**
- **Support des instructions C-SRAM, par des passes d'optimisation dans Hybrogen**

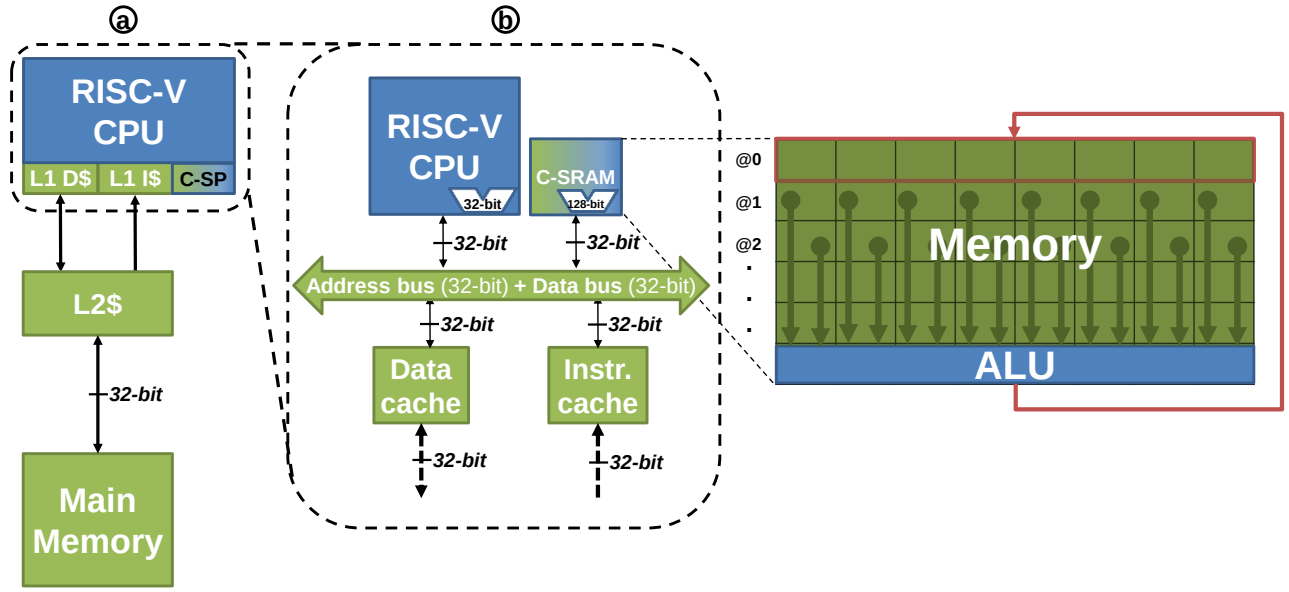


☆ *Mambu et al., Instruction Set Design Methodology for In-Memory Computing through QEMU-based System Emulator. ESWEEK RSP 2021.*

Contribution – Plateforme de compilation et de simulation

- **Validation des résultats : évaluation de trois applications C-SRAM sur une architecture HPC**

- **Métriques :**
  - **Taux d'accélération / de réduction énergétique**
  - **Reduction des misses de lecture / écriture sur les caches CPU**



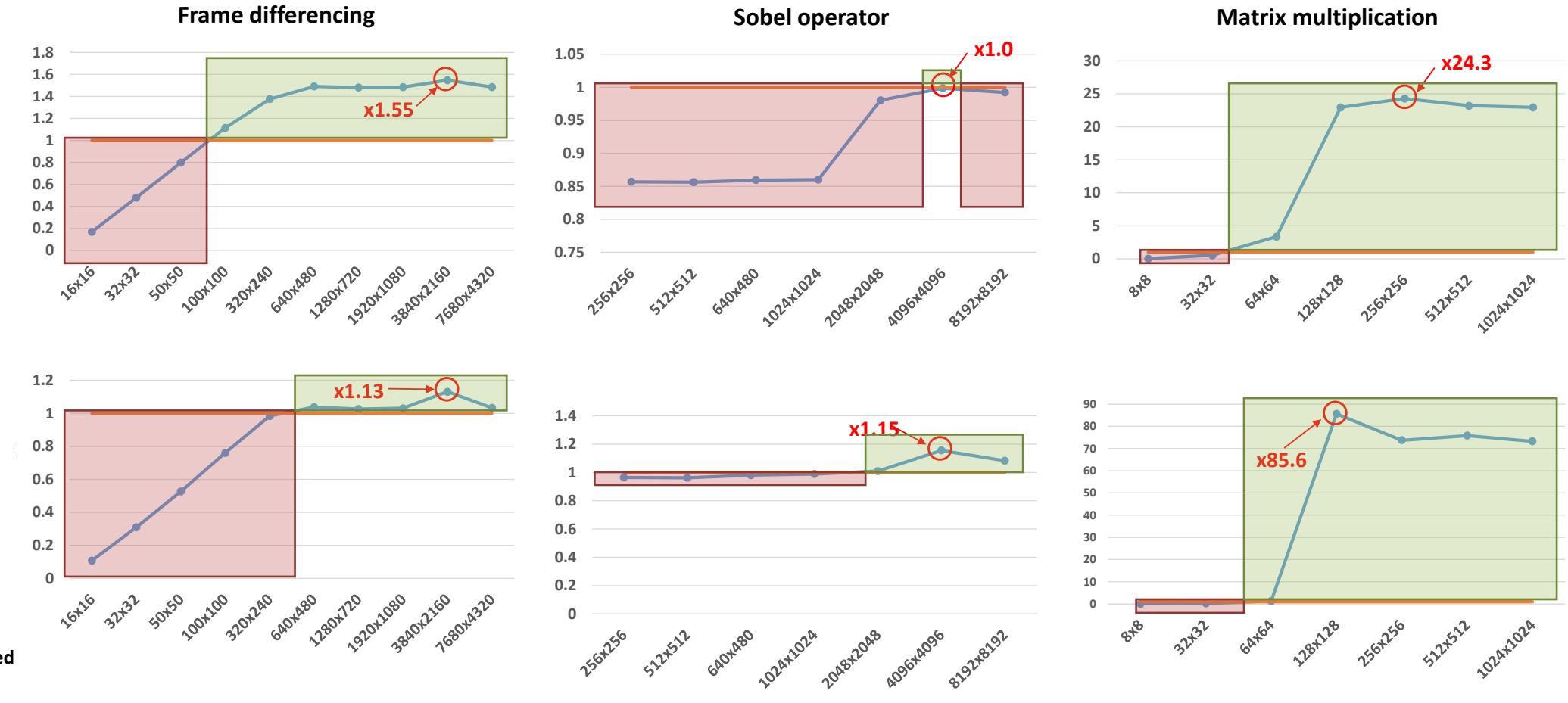
Application	Taille de données	Complexité	Motif accès mémoire	Facteur de redondance données
Différence d'image	8-bit	$O(n)$	Row-major	1
Filtre de Sobel	16-bit	$O(n)$	Stencil	≈ 18
Multiplication matricielle	32-bit	$O(n^3)$	Row-major / Column-major	n^2

☆ *Mambu et al., Instruction Set Design Methodology for In-Memory Computing through QEMU-based System Emulator. ESWEEK RSP 2021.*

Contribution – Plateforme de compilation et de simulation – Résultats

Taux d'accélération

Taux de réduction énergétique

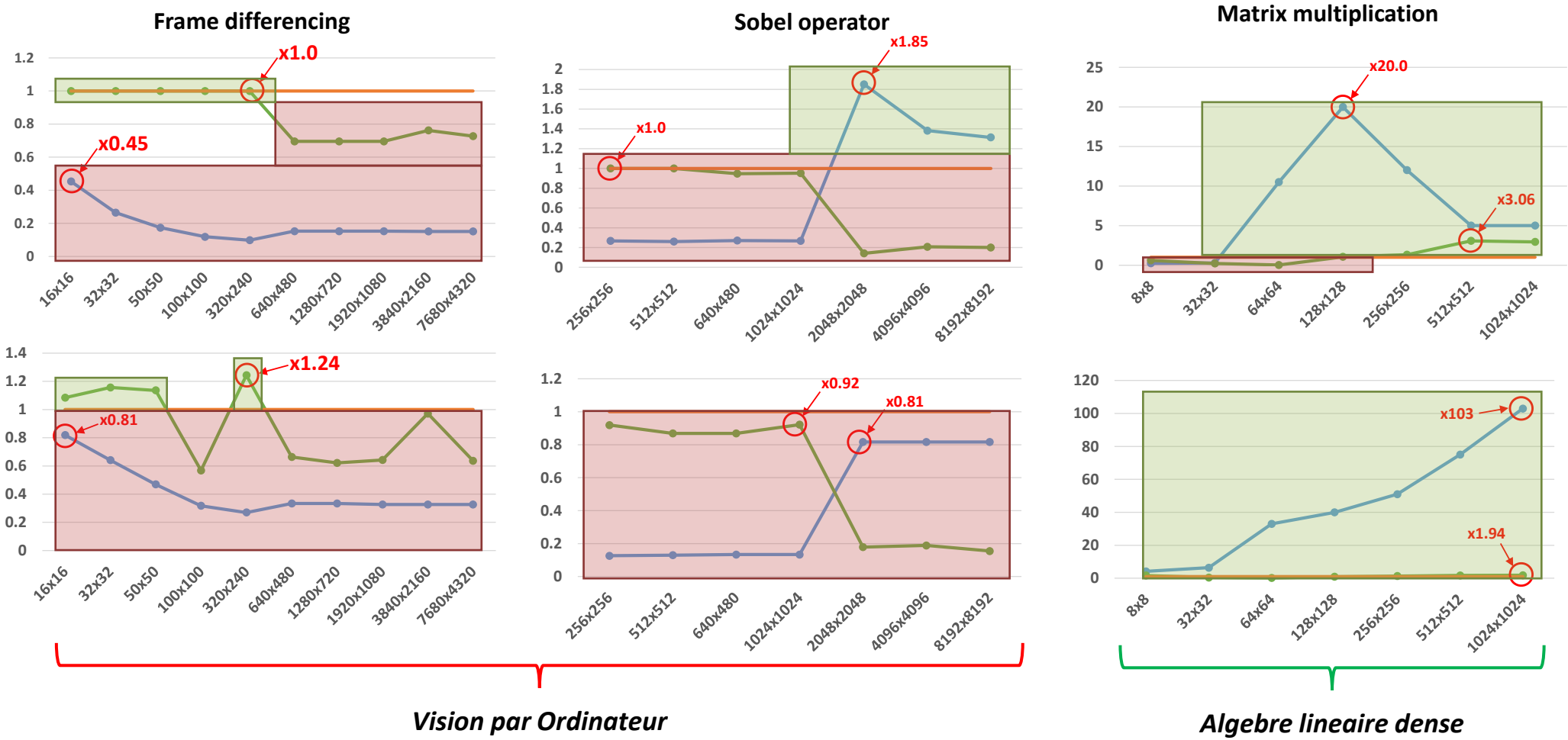


- Gains de performance mitigés pour la différence d'image ➔ intensité arithmétique faible
- Multiplication matricielle : gains importants ➔ tuilage de boucle + vectorisation de l'application
- Filtre de Sobel : gains de performance limitées ➔ gains de performance limités

Contribution – Plateforme de compilation et de simulation – Résultats

Reduction des misses de lecture

Reduction des misses d'écriture



- Tuilage de boucle optimise au maximum la consommation de bande passante
- Pour les deux autres applications : consommation de la bande passante inefficace ➔ **surcout logiciel de la gestion des transferts par le CPU, chemin de données inefficace sans dispositif supplémentaire**

Contribution – Plateforme de compilation et de simulation

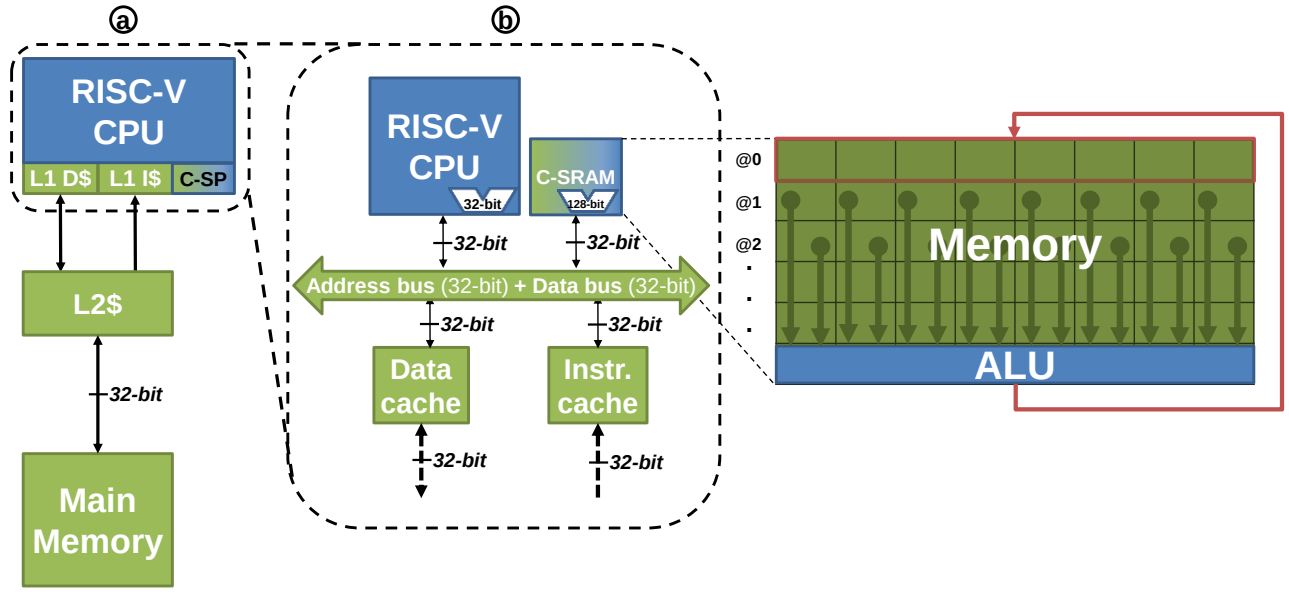
- **Validation des résultats : évaluation de trois applications C-SRAM sur une architecture HPC**

- **Métriques :**
  - **Gains de performance (Speed-up / Reduction énergétique)**
  - **Reduction des misses de lecture / écriture sur les caches CPU**

- **Conclusion :**

- **Intérêt du tuilage de boucle pour des applications à complexité cubique**

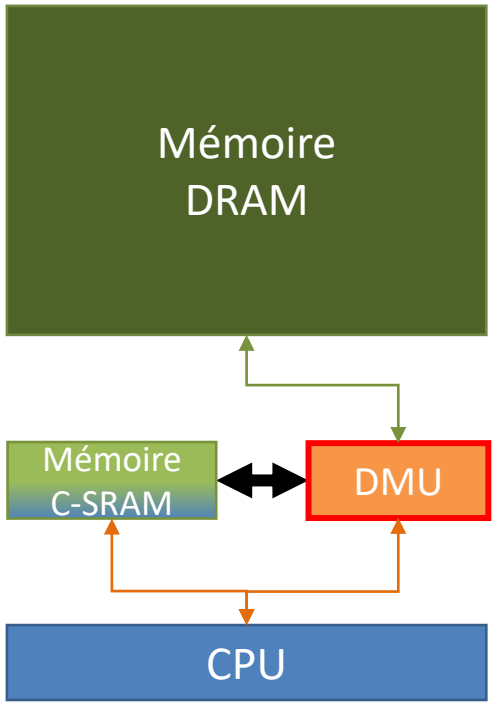
➔ **Intérêt d'une meilleure interface entre la mémoire C-SRAM et la mémoire DRAM**



Application	Taille de données	Complexité	Motif accès mémoire	Facteur de redondance données
Différence d'image	8-bit	$O(n)$	Row-major	1
Filtre de Sobel	16-bit	$O(n)$	Stencil	≈ 18
Multiplication matricielle	32-bit	$O(n^3)$	Row-major / Column major	n^2

☆ *Mambu et al., Instruction Set Design Methodology for In-Memory Computing through QEMU-based System Emulator. ESWEEK RSP 2021.*

Dispositif intelligent pour le transfert de données de stencil



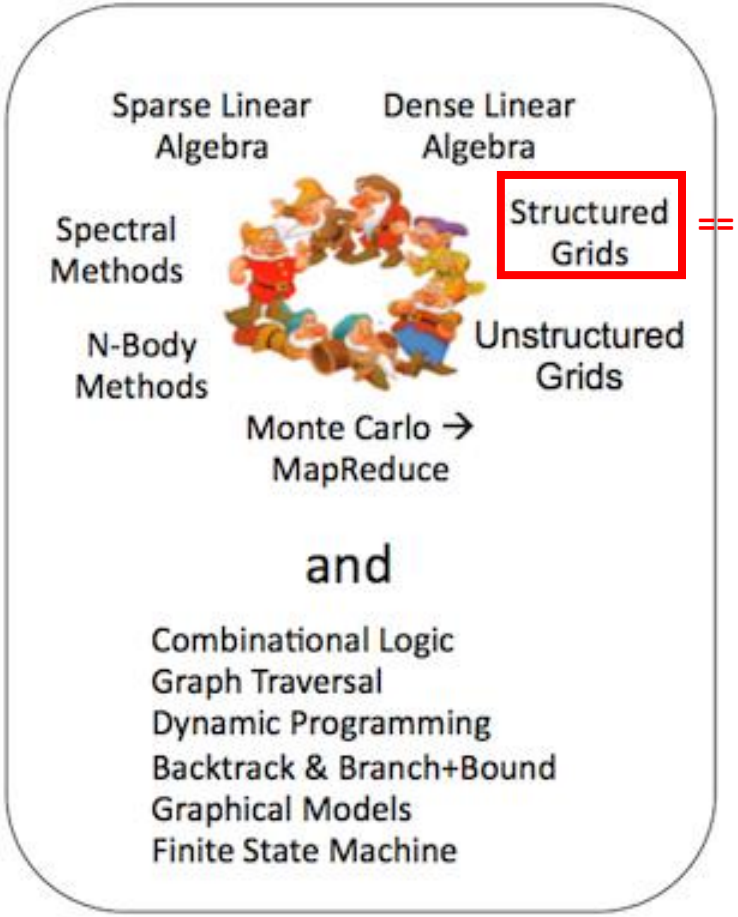
- *Transferts de données de bloc (type DMA)*
- *Transferts de données spéciaux et paramétrables*

- *Flot de contrôle & synchronisation*

- **<u>Data-locality Management Unit (DMU)</u> pour le transfert de données entre mémoire DRAM et C-SRAM.**
- **Réorganisation de données pour permettre la vectorisation des calculs au sein de la mémoire C-SRAM.**

☆ Mambu et al., Towards Integration of a Dedicated Memory Controller and Its Instruction Set to Improve Performance of Systems Containing Computational SRAM. MDPI JLPEA 2022.

Motivations logicielles – Calculs de stencils



== calcul de stencil

Les “dwarfs”, 13 motifs de programmation génériques retrouvés dans le paysage informatique [4].

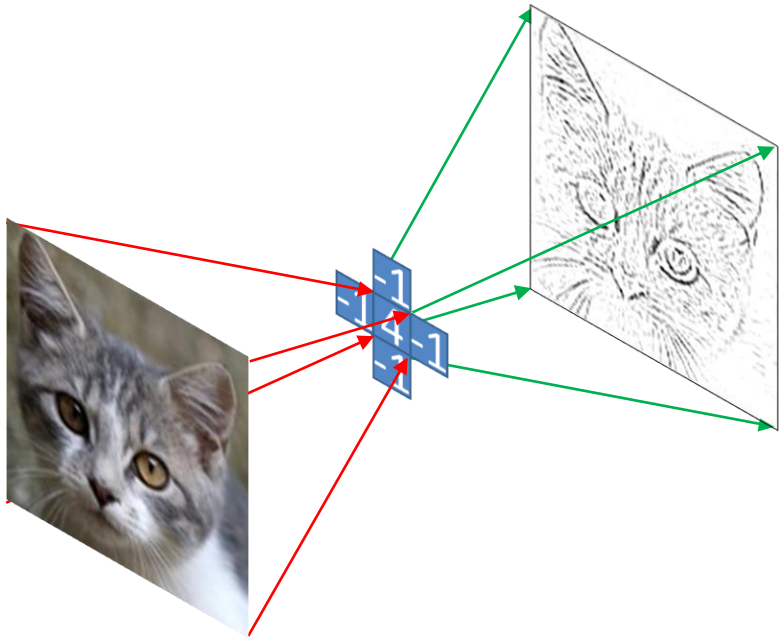
OpenCV

OpenVX™

Forte présence de stencils dans les spécifications de Vision par Ordinateur (ex : OpenCV & OpenVX)

[4] Feng et al., OpenCL and the 13 Dwarfs: A Work in Progress, 2012.

Motivations logicielles – Calculs de stencils

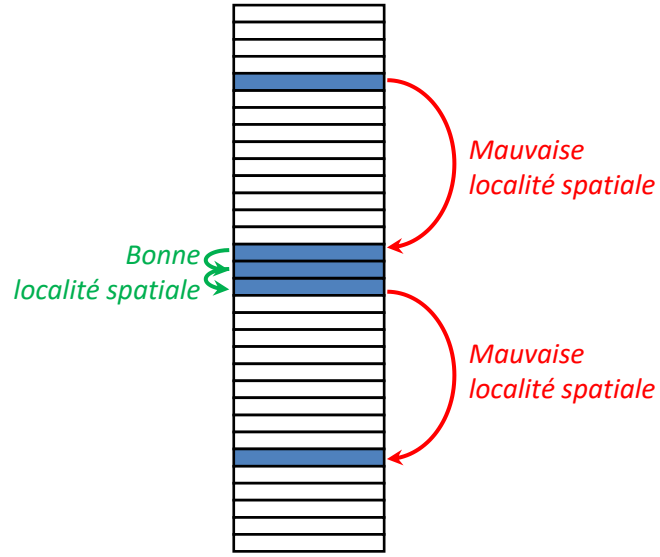


Exemple : le Laplacien discret, à 5 points.

```
void Laplacian_5pt(char Input[N][N], char
Output[N][N])
{
  for (int i = 1; i < N-1; i += 1)
  {
    for (int j = 1; j < N-1; j += 1)
    {
      Output[i][j] = -1 * Input[i-1][j  ]
                     -1 * Input[i  ][j-1]
                     +4 * Input[i  ][j  ]
                     -1 * Input[i  ][j+1]
                     -1 * Input[i+1][j  ];
    }
  }
}
```

Intensité arithmétique :
(5 Mul + 4 Add) / 5 Octets = 1,8 IntOps / Octet

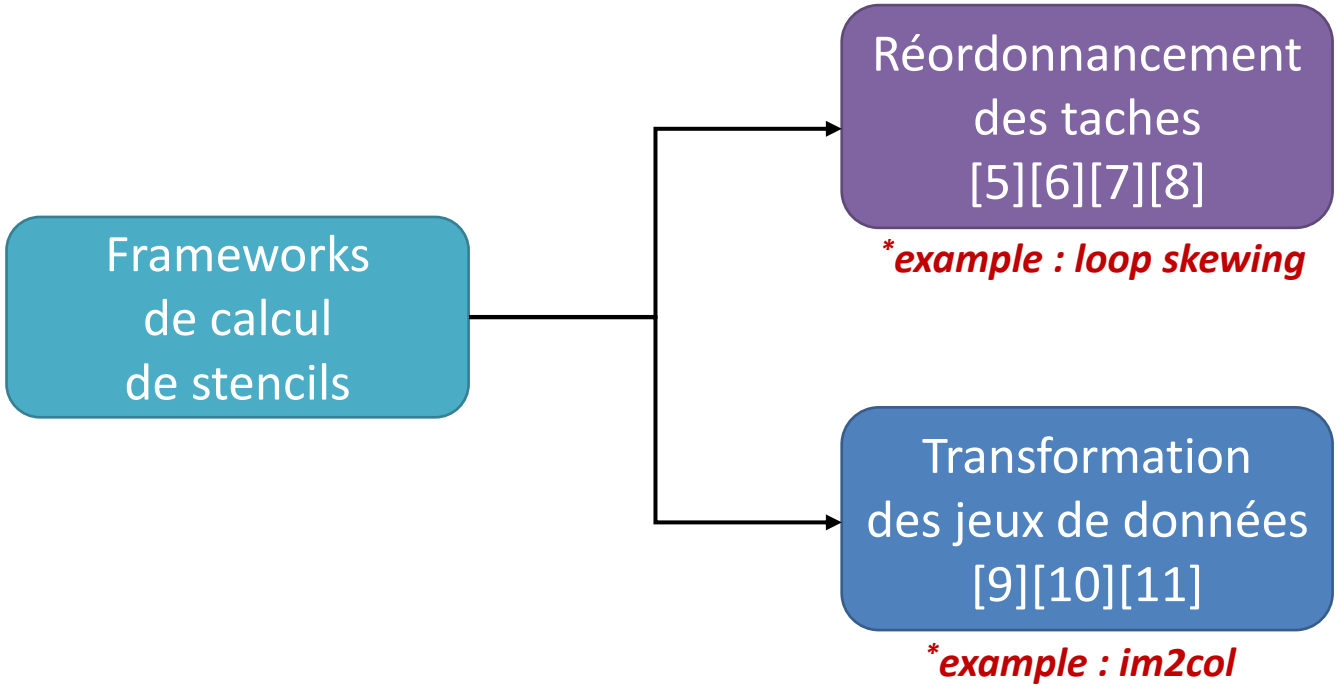
Implémentation algorithmique, en langage C.



Motif d'accès en mémoire.

- Principe fondamental : appliquer sur chaque point d'une grille ordonnée (2D ou 3D) une fonction discrète (approximée)
- Permet d'effectuer des opérations complexes avec une complexité arithmétique moindre.
- Contre-coup : très intensif en accès mémoire → Sensible au goulot d'étranglement de von Neumann.

Calculs de stencils – Etat de l'Art – Support logiciel



[+] Optimisation algorithmique → gains de performance garantis

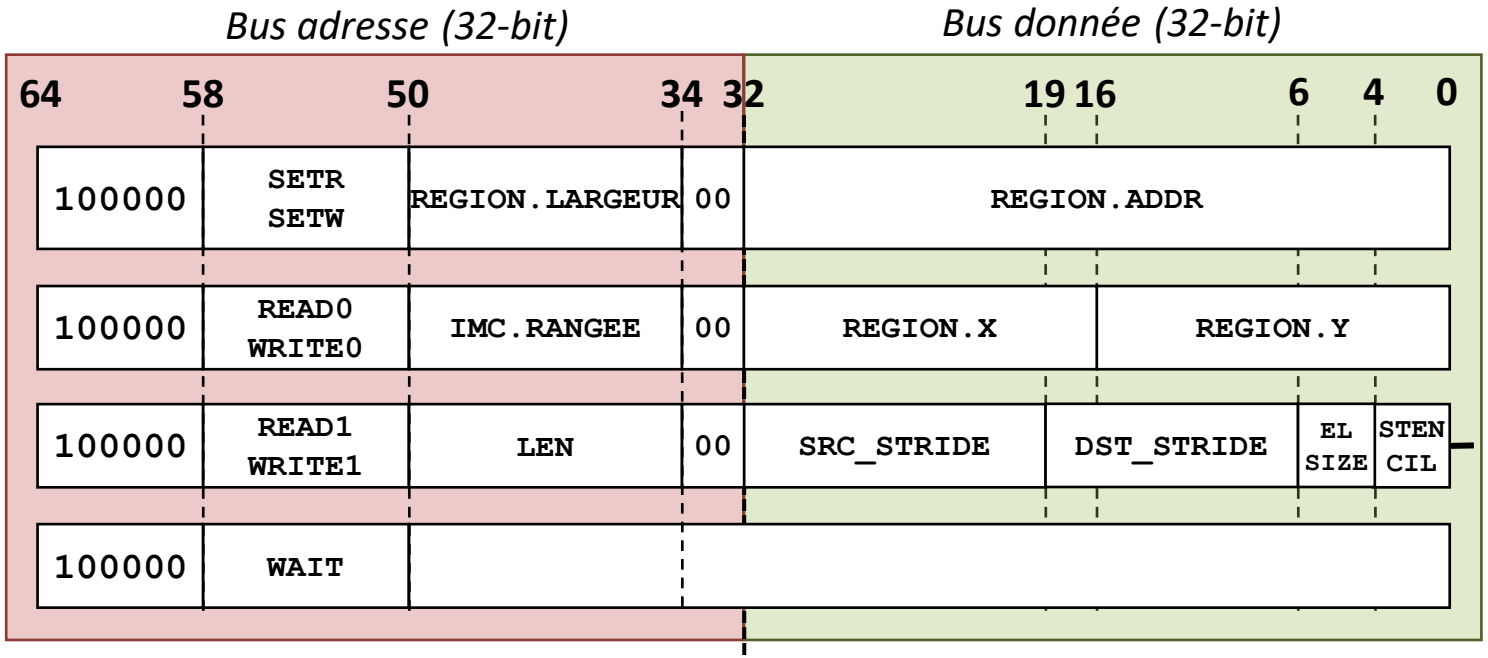
[-] Peut être perturbé par des fonctions matérielles non-programmable (ex : politiques de caches, prefetchers)

[+] Meilleure utilisation de la bande passante pour les architectures Haute Perf.

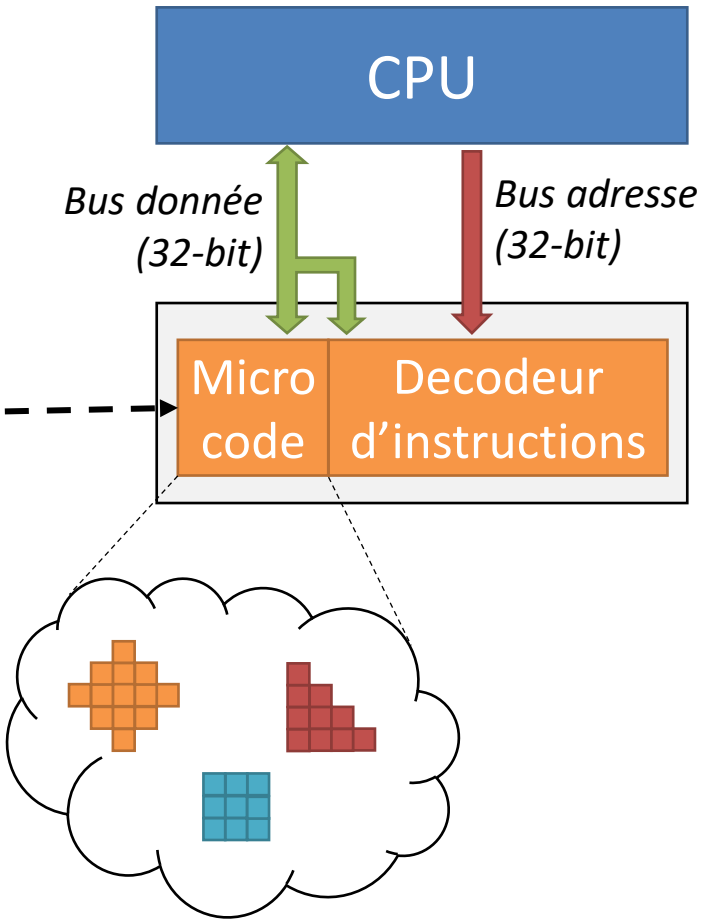
[-] Augmentation de l'empreinte mémoire → aggravation de l'Energy Wall

[5] Lacassagne et al., High-Level Transforms For SIMD and Low-Level Computer Vision algorithms. Proceedings of WPMVP 2014.
[6] Ragan-Kelley et al., Halide : Decoupling Algorithms from Schedules for High-Performance Image Processing. ACM Communications 2017.
[7] Hegarty et al., Darkroom : compiling high-level image processing code into hardware pipelines. ACM Trans. Graph. 2014.
[8] Mullapudi et al., Polymage : Automatic Optimization for Image Processing Pipelines. SCM SIGARCH 2015.
[9] Lu et al., Data Layout Transformation for Enhancing Data Locality on NUCA Chip Multiprocessors. IEEE PACT 2009.
[10] Zhang et al., Data Layout Transformation for Stencil Computations Using ARM NEON Extension. IEEE HPCC 2020.
[11] Henretty et al., Data Layout Transformation for Stencil Computations on Short-Vector SIMD Architectures. CC 2011.

Contribution – Jeu d'instructions pour le transfert de stencils



Jeu d'instructions du DMU (programmé par le logiciel)



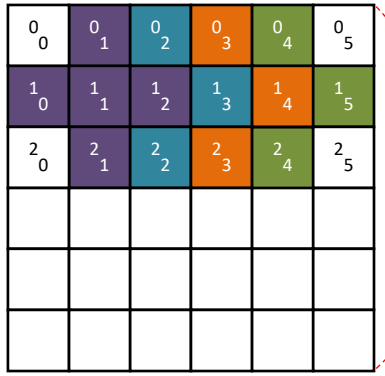
Stencils de l'application (fournis par le logiciel)

- **Jeu d'instructions pour programmer les transferts de stencils, de forme arbitraire [28].**
- **Utilisation d'un micro-code pour programmer les stencils et optimiser les transferts redondants [29].**

☆ Mambu Charles et Kooli – Système de transfert direct de données #1. Brevet déposé à l'INPI.
☆ Mambu Charles et Kooli – Système de transfert direct de données #2. Brevet déposé à échelle internationale

Contribution – Jeu d'instructions pour le transfert de stencils

Transfert vers…



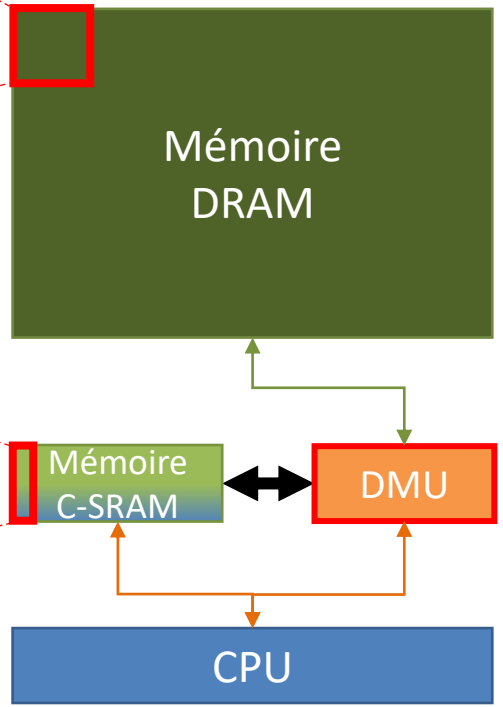
Zoom sur quatre stencils '+' à transférer.

Image d'entrée

-1	-1	-1	-1	-1	-1	-1	-1
4	4	4	4	4	4	4	4
0,1	0,2	0,3	0,4				
1,0	1,1	1,2	1,3				
1,1	1,2	1,3	1,4				
1,2	1,3	1,4	1,5				
2,1	2,2	2,3	2,4				
16x8-bit ALU							

Coefficients Laplacien discret

Les quatre stencils, transposés verticalement dans la mémoire C-SRAM, afin d'être calculés.

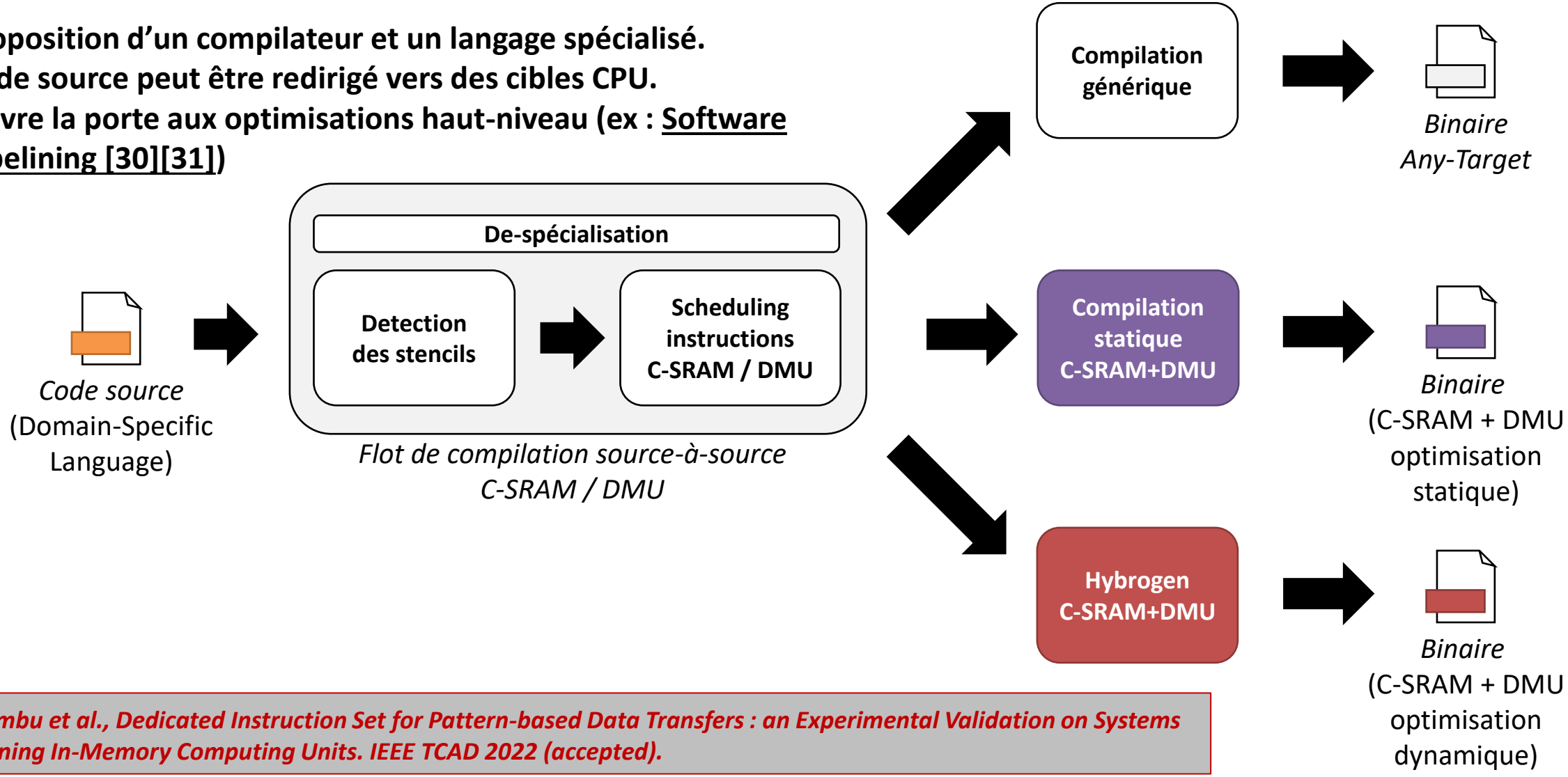


- *Transferts de données de bloc (type DMA)*
- ***Transferts de données de stencil***
- *Flot de contrôle & synchronisation*

☆ Mambu et al., Towards Integration of a Dedicated Memory Controller and Its Instruction Set to Improve Performance of Systems Containing Computational SRAM. MDPI JLPEA 2022.

Contribution – Flot de compilation pour la mémoire C-SRAM et le DMU

- **Proposition d'un compilateur et un langage spécialisé.**
- **Code source peut être redirigé vers des cibles CPU.**
- **Ouvre la porte aux optimisations haut-niveau (ex : Software Pipelining [30][31])**



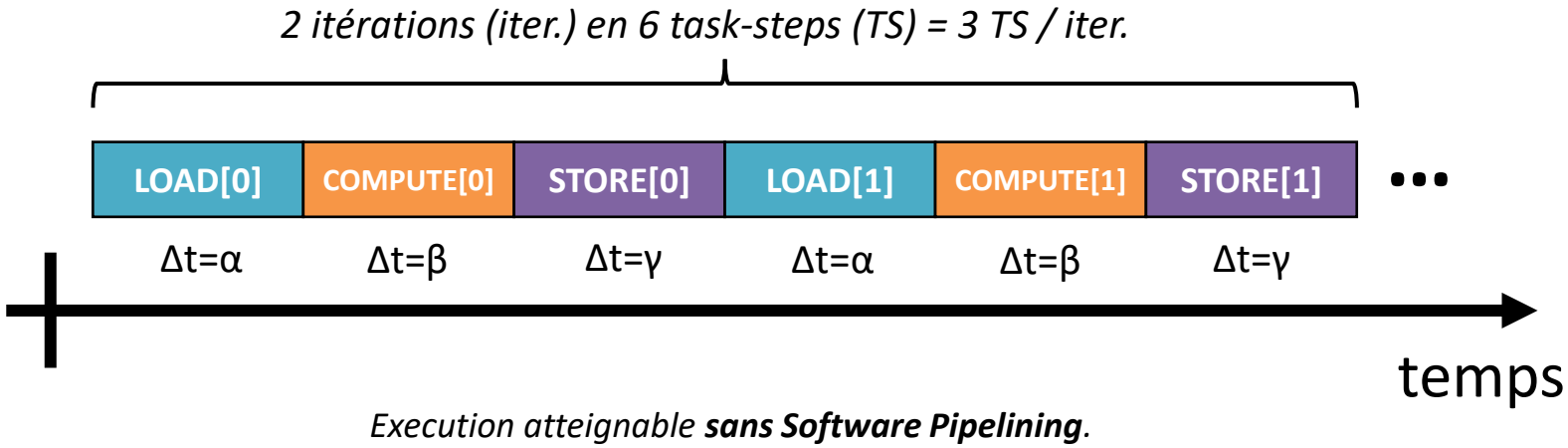
☆ ***Mambu et al., Dedicated Instruction Set for Pattern-based Data Transfers : an Experimental Validation on Systems Containing In-Memory Computing Units. IEEE TCAD 2022 (accepted).***

[31] Saidi et al., Optimizing explicit data transfers for data parallel applications on the cell architecture. ACM TACO 2012.

Contribution – Support logiciel du Data-locality Management Unit

```
int i = 0;
int tmpA, tmpB, tmpC;
for(i = 0; i < N; i += 1)
{
  // Deps de donnees
  // ➔ transferts bloquants
  tmpA = A[i];
  tmpB = B[i];
  // Computation
  tmpC = tmpA - tmpB;
  // Blocking output transfers
  // due to data dependency
  C[i] = tmpC;
}
```

Code d'exemple



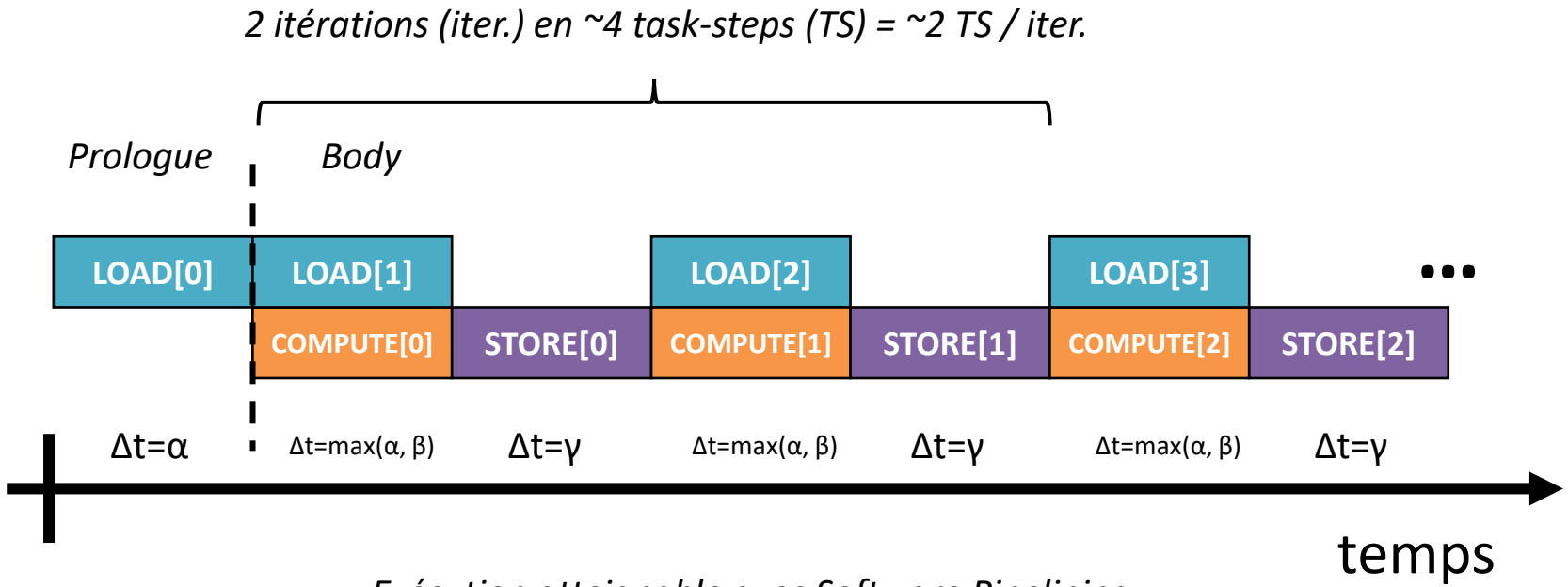
*Execution atteignable **sans Software Pipelining**.*

- **Soit un code composées de tâches « LOAD », « COMPUTE » et « STORE ».**
- **La dépendance de données entre tâches peut limiter les perspectives d'optimisation du code…**

Contribution – Support logiciel du Data-locality Management Unit

```
int i = 0;
int idx = 0, nidx = 1;
int tmpA[2], tmpB[2], tmpC[2];
// Prologue
tmpA[idx] = A[i];
tmpB[idx] = B[i];
// Main body
for (; i < N; i += 1)
{
  if (i < (N-1))
  {
    // Data-independent input
transfers
    tmpA[nidx] = A[i];
    tmpB[nidx] = B[i];
  }
  // Computation
  tmpC[idx] = tmpA[idx] - tmpB[idx];
  // Data-independent output
transfers
  C[i-1] = tmpC[idx];
  // Update of buffer indices
  swap(idx, nidx);
}
```

Code d'exemple



Exécution atteignable avec Software Pipelining
sur un bus de données mono-port.

- **Utilisation du Software Pipelining pour exploiter au maximum le recouvrement des accès mémoire et du calcul**

Etat des travaux scientifiques

Vision Par Ordinateur
2017

Applications

1. What are the applications?
2. What are common kernels of the applications?

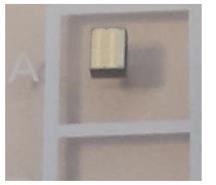
Flot de compilation
2022

Programming Models

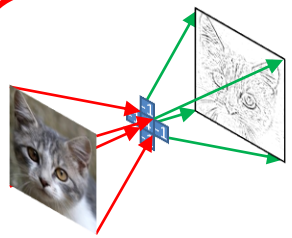
5. How to describe applications and kernels?
6. How to program the hardware?

Hardware

3. What are the hardware building blocks?
4. How to connect them?



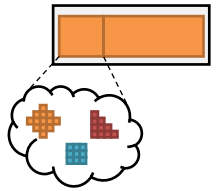
1er circuit-test
2020



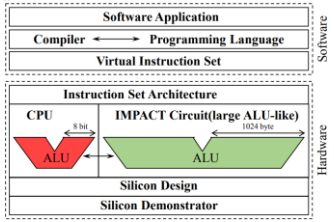
Codes de stencils
2020

Evaluation:

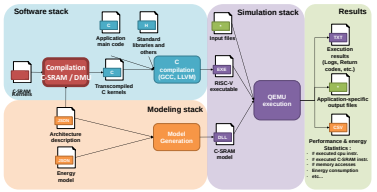
7. How to measure success?



Jeu d'instructions pour les stencils
2021



1er modèle de simulation
2017



Plateforme de simulation système
2020

Méthodologie expérimentale

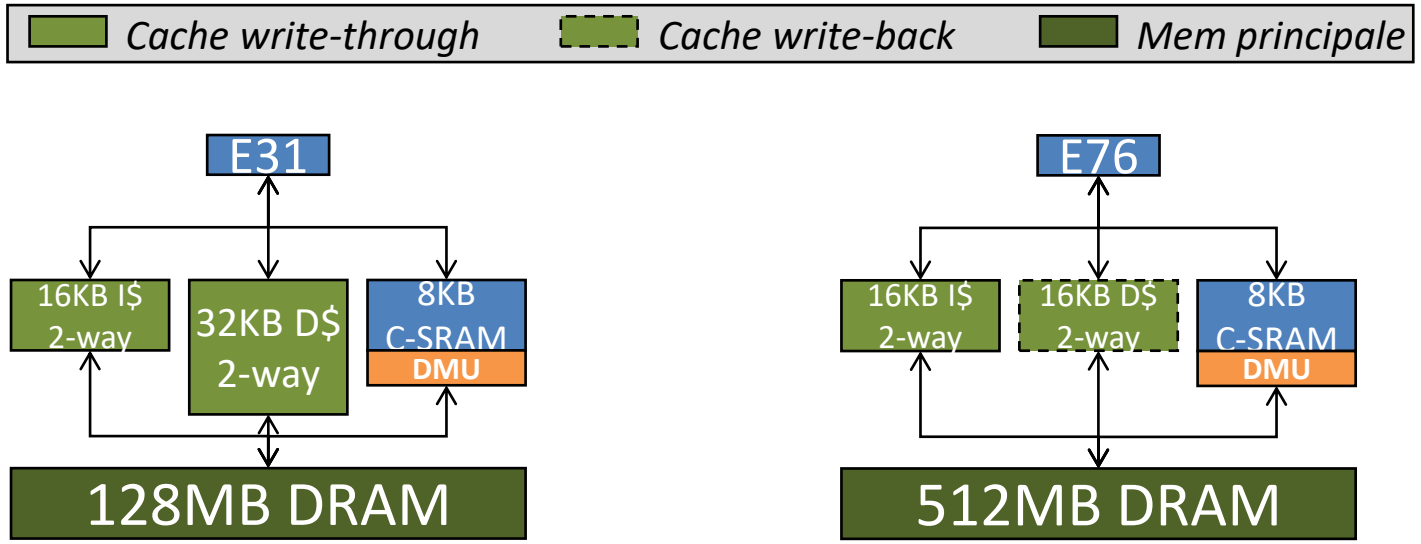
Objectifs :

- *Quels sont les gains de performance obtenus par l'utilisation de la C-SRAM et du DMU, à travers le modèle de programmation proposé ?*

Paramètres expérimentaux identifiés:

- Les applications implémentées
  - Pas de micro-benchmarking, pour évaluation au niveau système
- Les architectures de calcul
- Les métriques qualitatives
  - Métriques spécialisées autour du calcul de stencils (non-dépendantes de suites de benchmarks)

Méthodologie expérimentale – Architectures

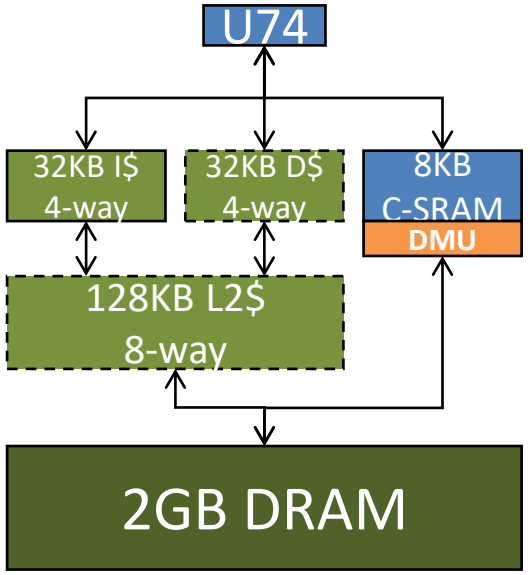


Système 1 (384MHz) [38]

Component	Latency (cycles)	Energy cost
C-SRAM scratchpad	1	31.74pJ
L1 D$, 2-way • **write-through policy**	1	Read: 19pJ Write: 25pJ
L1 I$, 2-way • **write-through policy**	1	Read: 34pJ Write: 34pJ
DRAM memory	7	Read: 8.17nJ Write: 8.04nJ

Système 2 (800MHz) [39]

Component	Latency (cycles)	Energy cost
C-SRAM scratchpad	2	31.74pJ
L1 D$, 4-way • **write-back policy**	1	Read: 19pJ Write: 25pJ
L1 I$, 2-way • **write-through policy**	1	Read: 34pJ Write: 34pJ
DRAM memory	24	Read: 14.45nJ Write: 14.35nJ

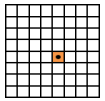
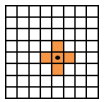
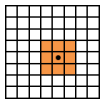
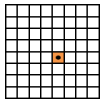
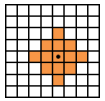


Système 3 (1GHz) [40]

Component	Latency (cycles)	Energy cost
C-SRAM scratchpad	3	31.74pJ
L1 D$, 4-way • **write-back policy**	1	Read: 24pJ Write: 24pJ
L1 I$, 4-way • **write-through policy**	1	Read: 24pJ Write: 24pJ
L2 $, 8-way • **write-back policy**	12	Read: 52pJ Write: 52pJ
DRAM memory	48	Read: 39nJ Write: 37.5nJ

[38] SiFive, SiFive FE310-G000 Manual v2p3.
[39] Bob Wheeler – WD Rolls Its Own RISC-V Core. The Linley Group Microprocessor Report. Feb 2019.
[40] SiFive, SiFive HF105 Datasheet.

Méthodologie expérimentale – Applications

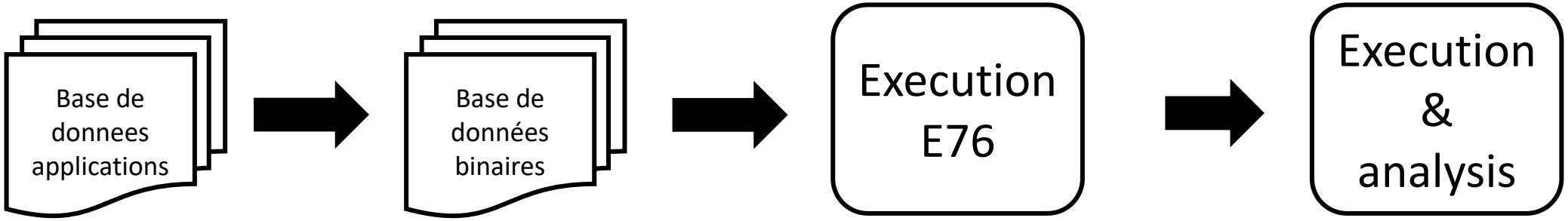
Application	Arithmétique	Complexité	Pattern	Facteur de redondance données
Différence d'image [33]	16-bit	O(n)		1
Laplacien Discret [33][34]	16-bit	O(n)		~5
Filtre de Sobel [34][35]	16-bit			~18
Multiplication Matricielle [36]	32-bit	O(n^3)		n^2
Dématriçage [37]	16-bit	O(n)		~26

[33] Nishu Singhla, Motion Detection Based on Frame Difference Method. IJICT 2014.
[34] The OpenCV Specification. https://opencv.org
[35] The OpenVX Specification. https://khronos.org/openvx
[36] Basic Linear Algebra Subprograms (BLAS) Technical Forum Report. 2001.
[37] Malvar He et Cutler, High-Quality Linear Interpolation for Demosaicing of Bayer-patterned Color Images. IEEE ASSP 2004.

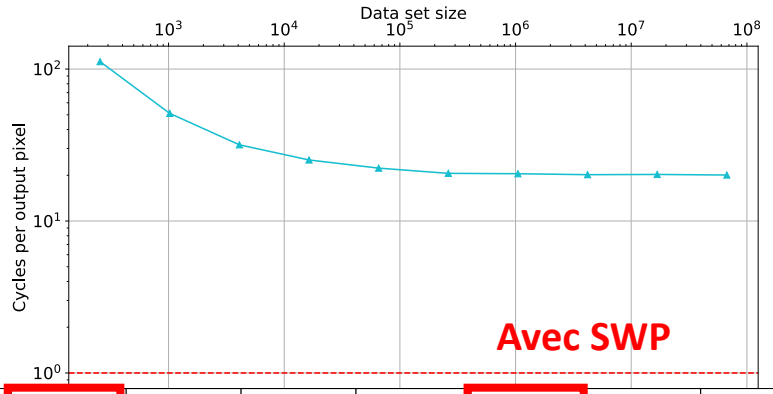
Expérience #1 – Impact du DMU sur la performance architecturale de la C-SRAM

- *Difference d'image*
- *Laplacien discret*
- *Filtre de Sobel*
- *Multiplication matricielle*
- *Dématriçage*

- Version CPU pour ref. : 1 version
- In/Out CPU/DMU? : 4 versions
- Naïf? SWP ? : 2 versions
- **Total : 9 versions par app.**
- Size set: 10 input sizes
- **Total : 90 runs par app.**

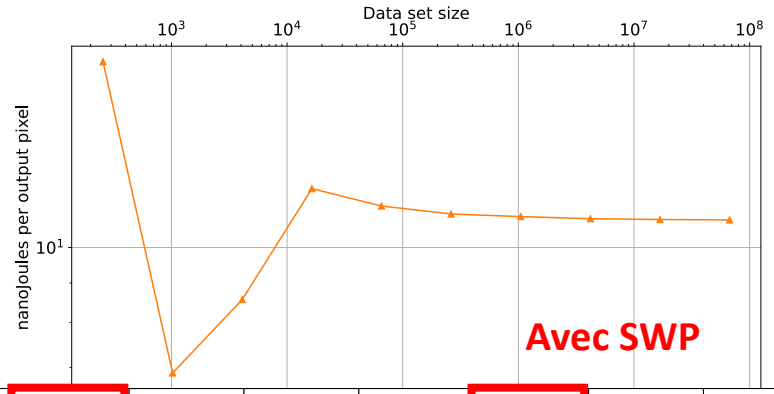


Expérience #1 – Impact du DMU sur la performance architecturale de la C-SRAM



	risc-v, naive	DD, naive	DC, naive	CD, naive	CC, naive	DD, db	DC, db	CD, db	CC, db
256 B	12.27	166.957	156.699	148.23	145.383	130.293	118.43	115.133	111.805
1 KB	11.083	89.099	85.144	70.006	74.018	62.716	57.344	51.073	53.717
4 KB	21.431	53.099	56.02	42.176	53.704	35.882	38.289	31.732	43.945
16 KB	39.643	36.814	46.629	39.387	55.738	25.222	34.564	34.018	50.298
64 KB	34.601	28.014	37.443	28.509	43.794	22.288	28.303	26.312	42.215
256 KB	36.653	23.499	32.692	23.54	39.175	20.593	25.114	22.273	38.075
1 MB	58.642	21.397	30.645	21.29	36.968	20.474	26.039	22.983	40.144
4 MB	58.385	21.151	30.348	21.131	37.094	20.18	24.98	21.881	38.563
16 MB	58.16	21.04	30.219	22.508	36.907	20.252	25.024	21.828	38.492
64 MB	58.072	20.781	29.95	21.398	36.617	20.077	24.752	21.572	38.102

Sans SWP *Cycle Par Points (CPP)*



	risc-v, naive	DD, naive	DC, naive	CD, naive	CC, naive	DD, db	DC, db	CD, db	CC, db
256 B	3.86e+00	2.71e+01	2.26e+01	2.27e+01	1.74e+01	3.28e+01	2.84e+01	2.67e+01	2.13e+01
1 KB	3.17e+00	1.56e+01	1.36e+01	9.13e+00	6.96e+00	1.66e+01	1.40e+01	9.74e+00	6.88e+00
4 KB	9.14e+00	1.27e+01	1.39e+01	8.82e+00	1.06e+01	1.26e+01	1.36e+01	8.57e+00	1.08e+01
16 KB	1.97e+01	1.19e+01	1.66e+01	1.51e+01	1.95e+01	1.19e+01	1.65e+01	1.49e+01	1.92e+01
64 KB	1.68e+01	1.13e+01	1.56e+01	1.28e+01	1.64e+01	1.13e+01	1.56e+01	1.29e+01	1.70e+01
256 KB	1.80e+01	1.11e+01	1.50e+01	1.19e+01	1.57e+01	1.10e+01	1.51e+01	1.19e+01	1.58e+01
1 MB	3.07e+01	1.10e+01	1.49e+01	1.16e+01	1.54e+01	1.10e+01	1.57e+01	1.23e+01	1.70e+01
4 MB	3.05e+01	1.09e+01	1.48e+01	1.15e+01	1.54e+01	1.10e+01	1.54e+01	1.20e+01	1.64e+01
16 MB	3.04e+01	1.09e+01	1.47e+01	1.14e+01	1.53e+01	1.10e+01	1.54e+01	1.19e+01	1.63e+01
64 MB	3.03e+01	1.09e+01	1.47e+01	1.14e+01	1.53e+01	1.09e+01	1.53e+01	1.19e+01	1.62e+01

Sans SWP *nanoJoules Par Points (nJPP)*

Performance de la différence d'image pour l'architecture E76.

Expérience #1 – Impact du DMU sur la performance architecturale de la C-SRAM

Taille de donnees	256 B	1 KB	4 KB	16 KB	64 KB	256 KB	1 MB	4 MB	16 MB	64 MB
Sans SWP	166,96	89,10	53,10	36,81	28,01	23,50	21,40	21,15	21,04	20,78
Avec SWP	130,29	62,72	35,88	25,22	22,23	20,59	20,47	20,18	20,25	20,08
Gain	28,14%	42,07%	47,98%	45,96%	26,03%	14,11%	4,51%	4,81%	3,89%	3,51%

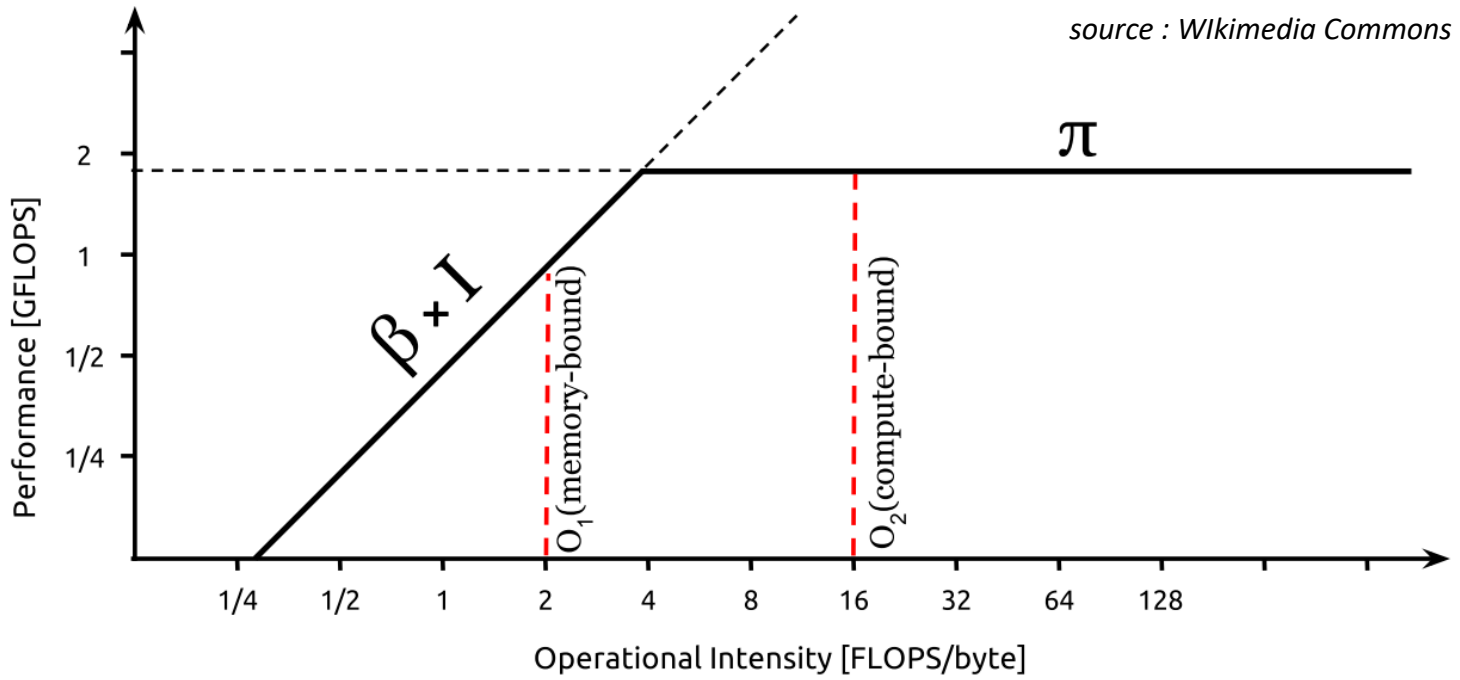
Cycles Par Point (CPP). Différence d'image, E76.

Taille de donnees	256 B	1 KB	4 KB	16 KB	64 KB	256 KB	1 MB	4 MB	16 MB	64 MB
Sans SWP	27,1	15,6	12,7	11,9	11,3	11,1	11	10,9	10,9	10,9
Avec SWP	32,8	16,6	12,6	11,9	11,3	11	11	11	11	10,9
Gain	-17,38%	-6,02%	0,79%	0,00%	0,00%	0,91%	0,00%	-0,91%	-0,91%	0,00%

nanoJoules Par Point (nJPP). Différence d'image, E76.

- **Gain de performance consistant pour toutes les tailles de code**
- **Surcout énergétique du SWP inferieur à 1% pour tailles de données > 4 KB ➔ SWP amorti !**

Expérience #2 – Exploitation logicielle de la C-SRAM et du DMU



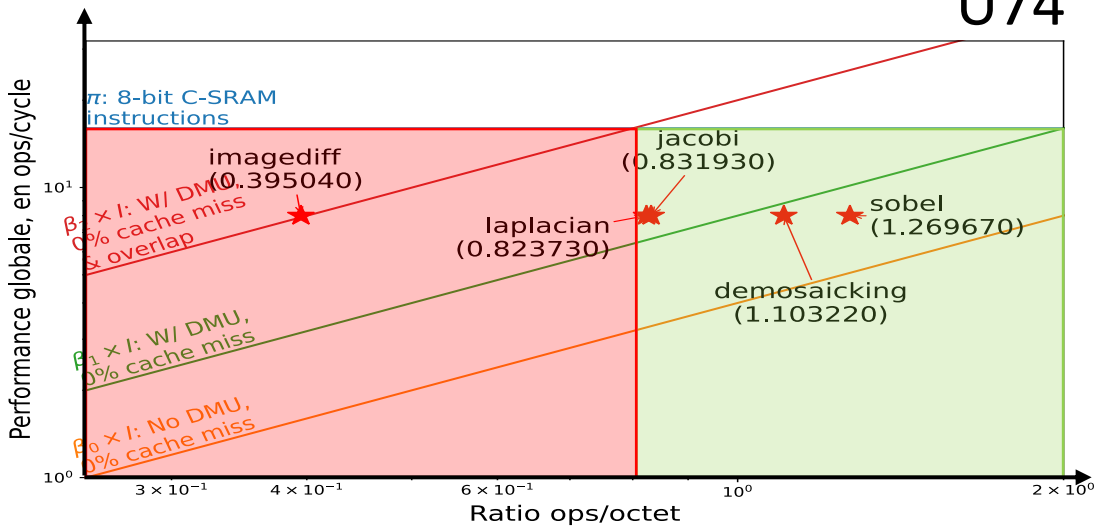
- Méthodologie d'évaluation basée sur le « *Roofline Model* » [41]
- Objectif : évaluer la qualité du code produit par le flot de compilation sur les architectures expérimentales.
  - Possible grâce a la distinction instructions entières CPU (Flot de contrôle) et instructions C-SRAM (calcul pratique)
- Effectuées pour les trois architectures (E31, E76, U74), avec les meilleures versions de code (noSWP / SWP).

[41] Williams Waterman et Patterson, Roofline : An Insightful Visual Performance Model For Multicore Architectures. Communications of the ACM #52. 2009.

Expérience #2 – Exploitation logicielle de la C-SRAM et du DMU

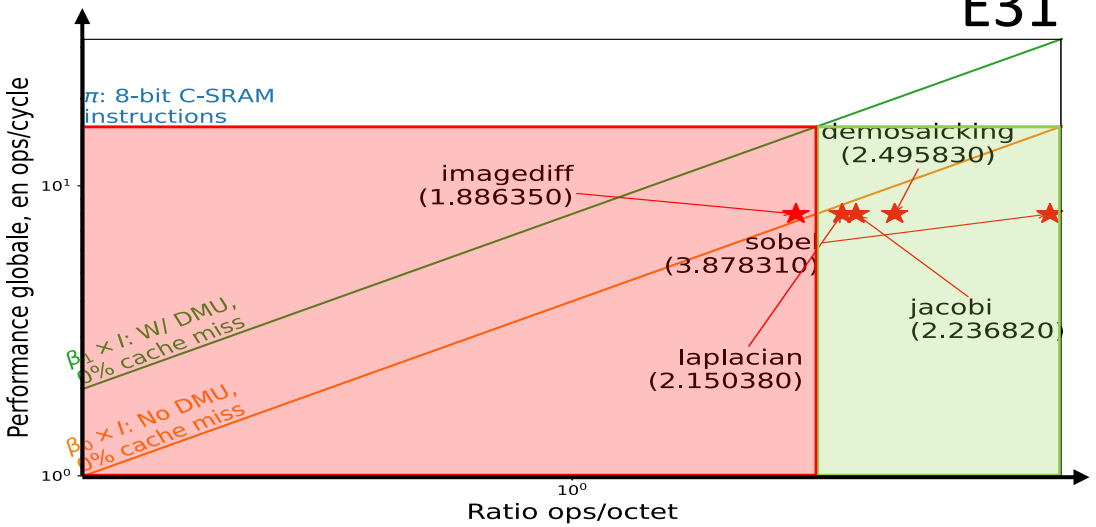
- **Comportement compute-bound pour la majorité des applications, sauf différence d'image (ratio ops/octet trop faible)**
- **Possibilité de quantifier les applications (16-bit➔8-bit) pour un débit de calcul plus important**

U74

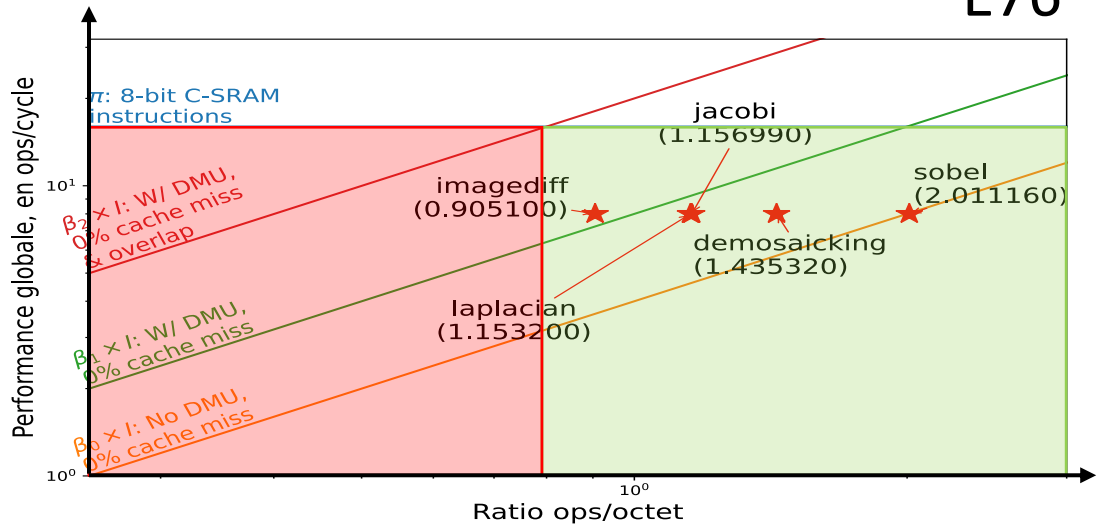


- **Ratios ops/octet limités sur l'architecture U74 (< 1.0)**
- **Limitation architecturale ➔ possibilité de repenser l'interface de la C-SRAM (ex : vers le cache L2 au lieu de la mémoire DRAM)**

E31



E76



Limitation de bande passante (memory-bound)

Limitation arithmétique (compute-bound)

Perspectives

Dispositif de transfert de stencils

- Implémentation, test et validation sur circuit.

Modélisation et évaluation des systèmes :

- Comparaison avec des architectures CPU à extension SIMD.
- Modélisation de systèmes multicœurs multi-C-SRAM (pour accélérer l'exploration architecturale, en opposition a la modélisation RTL).

Support logiciel / applicatif de la C-SRAM :

- Recherche autour d'autres solutions de calcul numérique intensif sur la mémoire (ex : calcul de matrices creuses).
- Optimisation dynamique plus agressive pour la programmation de la mémoire C-SRAM.
- Support d'optimisations haut-niveau pour le calcul de stencils (ex : *High-Level Transforms, Optimisation polyédrique*).
- Automatisation du Software Pipelining pour la mémoire C-SRAM et le DMU.

Conclusion

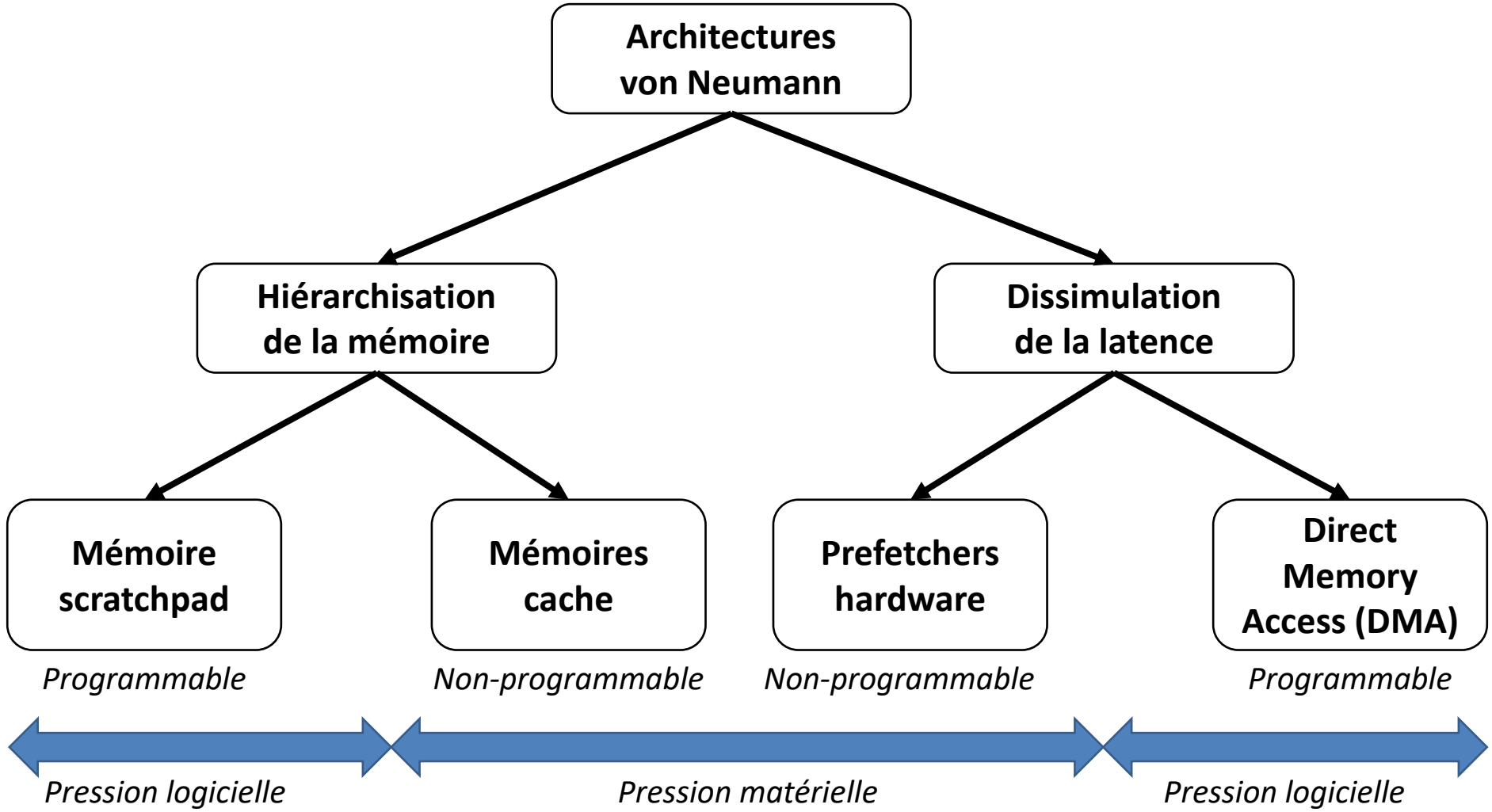
- **Spécifications du Data-locality Management Unit (DMU), un contrôleur mémoire pour transférer des stencils de la mémoire DRAM à la mémoire C-SRAM.**
- **Etablissement d'un modèle d'interface et de programmation pour le support logiciel de la mémoire C-SRAM et du DMU à haut-niveau.**
- **Proposition d'une méthodologie de modélisation et de simulation de systèmes intégrant la mémoire C-SRAM et le DMU.**
- **Evaluation expérimentale d'applications de stencils C-SRAM sur trois architectures expérimentales.**

Publications et brevets

- **Publications :**
- Julie Dumas, Henri-Pierre Charles, Kévin Mambu, Maha Kooli. ***Dynamic Compilation for Transprecision Applications on Heterogeneous Platform***. MDPI Journal of Low Power Electronics and Applications (JLPEA) 11, no. 3. Accepté et publié en Juin 2021.
- Maha Kooli, Antoine Heraud, Henri-Pierre Charles, Bastien Giraud, Roman Gauchi, Mona Ezzadeen, Kévin Mambu, Valentin Egloff, and Jean-Philippe Noël. ***Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution***. ACM Journal of Emerging Technologies and Computing Systems (JETC) 18, 2, Article 40. Accepté en Septembre 2021, publié en Avril 2022.
- Kevin Mambu, Julie Dumas, Henri-Pierre Charles, Maha Kooli. ***Instruction Set Design Methodology for In-Memory Computing through QEMU-based System Emulator***. 32nd International Workshop on Rapid System Prototyping (RSP). Accepté en Octobre 2021, publié en Juin 2022.
- Kévin Mambu, Henri-Pierre Charles, Maha Kooli, and Julie Dumas. ***Towards Integration of a Dedicated Memory Controller and Its Instruction Set to Improve Performance of Systems Containing Computational SRAM***. MDPI Journal of Low Power Electronics and Applications 12, no. 1 : 18. Accepté en Février 2022, publié en Mars 2022.
- Kévin Mambu, Henri-Pierre Charles, Maha Kooli. ***Dedicated Instruction Set for Pattern-based Data Transfers : an Experimental Validation on Systems Containing In-Memory Computing Units***. IEEE Transactions on Computer Design and Automation (TCAD). Accepté en Février 2023.

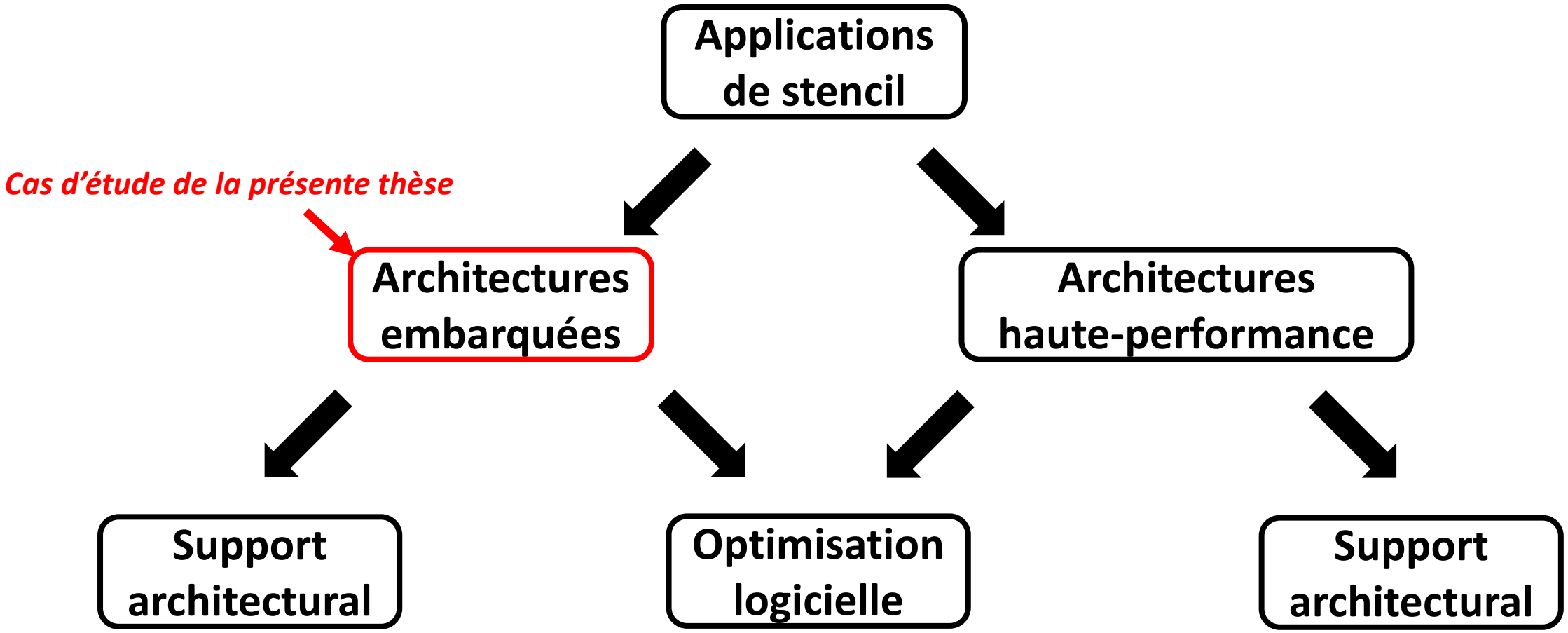
- **Brevets :**
- Kévin Mambu, Henri-Pierre Charles, Maha Kooli. ***Système de transfert direct de données #1***. Déposé à l'Institut National de la Propriété Intellectuelle (INPI).
- Kévin Mambu, Henri-Pierre Charles, Maha Kooli. ***Système de transfert direct de données #2***. Déposé à l'échelle internationale.

Modèle architectural de von Neumann – Mitigation du goulot d'étranglement

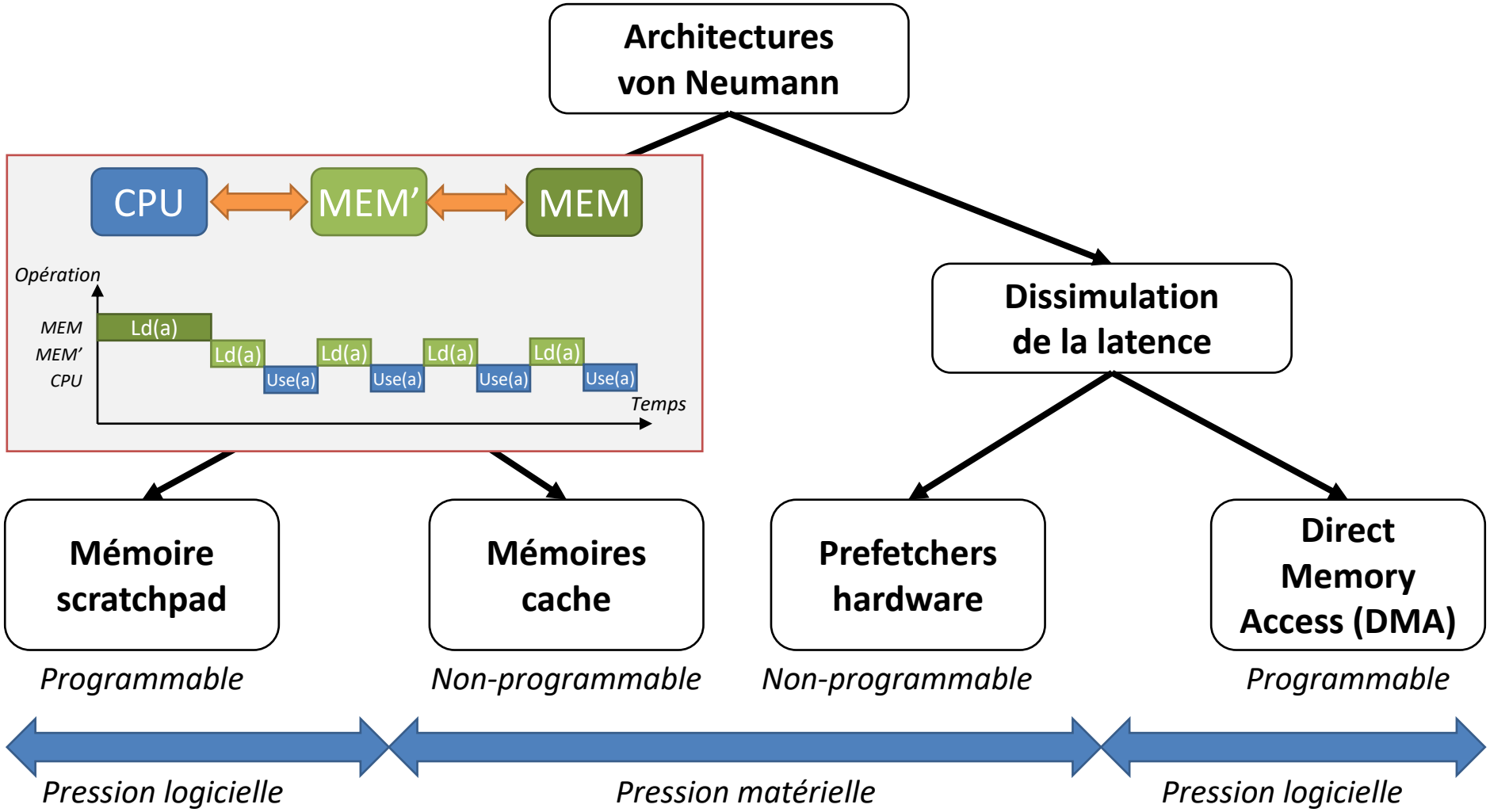


- *Hennessy & Patterson – A Quantitative approach, 5th edition, 2012*
- *Sparsh Mittal – A survey of recent prefetching techniques for processor caches, ACM Surveys (CSUR) 2016*

Calcul de codes de stencil – Etat de l'Art

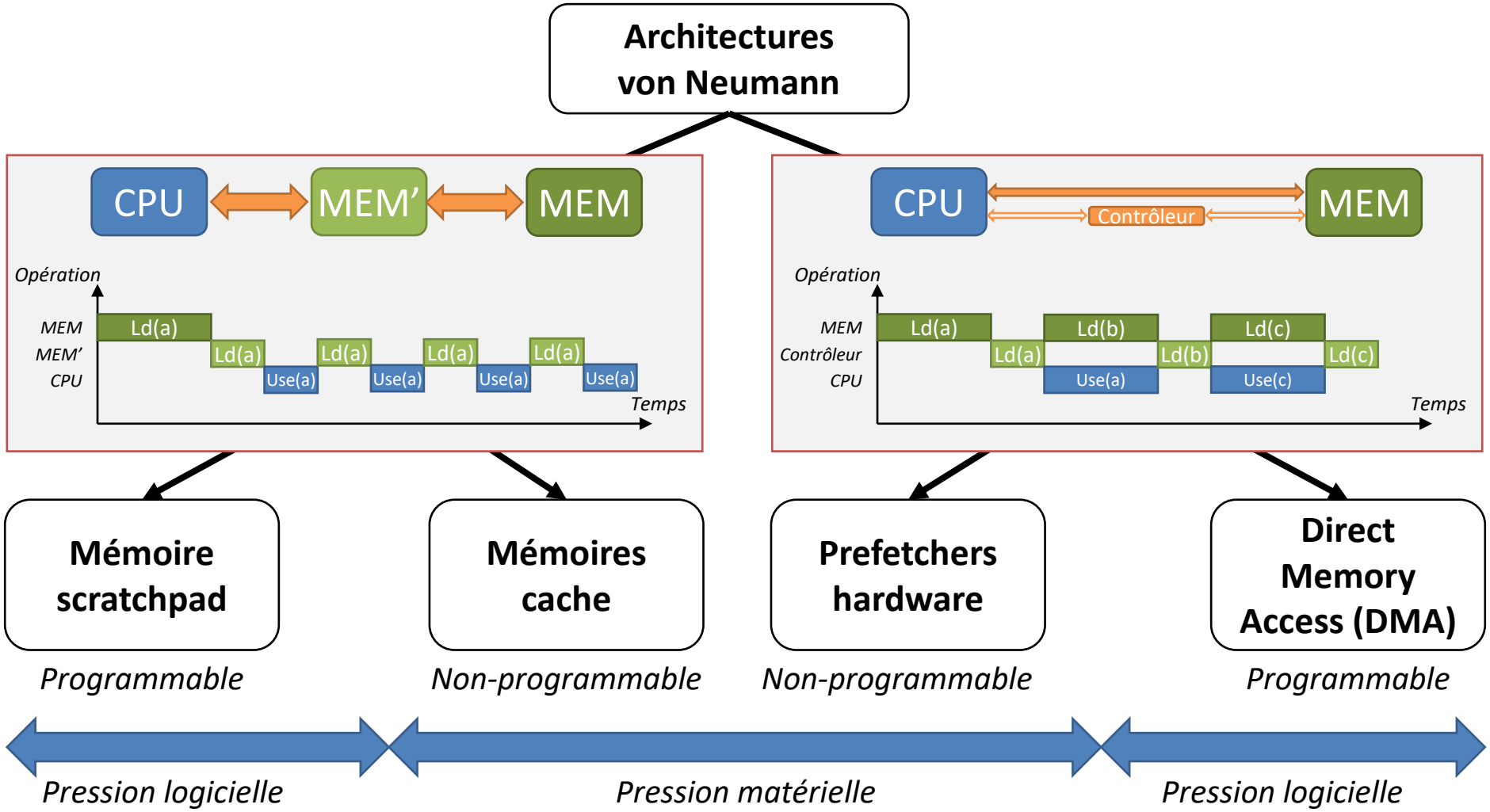


Modèle architectural de von Neumann – Mitigation du goulot d'étranglement



- *Hennessy & Patterson – A Quantitative approach, 5th edition, 2012*
- *Sparsh Mittal – A survey of recent prefetching techniques for processor caches, ACM Surveys (CSUR) 2016*

Modèle architectural de von Neumann – Mitigation du goulot d'étranglement



- *Hennessy & Patterson – A Quantitative approach, 5th edition, 2012*
- *Sparsh Mittal – A survey of recent prefetching techniques for processor caches, ACM Surveys (CSUR) 2016*

The Computing Continuum

IoT/Edge — Fog — HPC/Cloud/Instrument

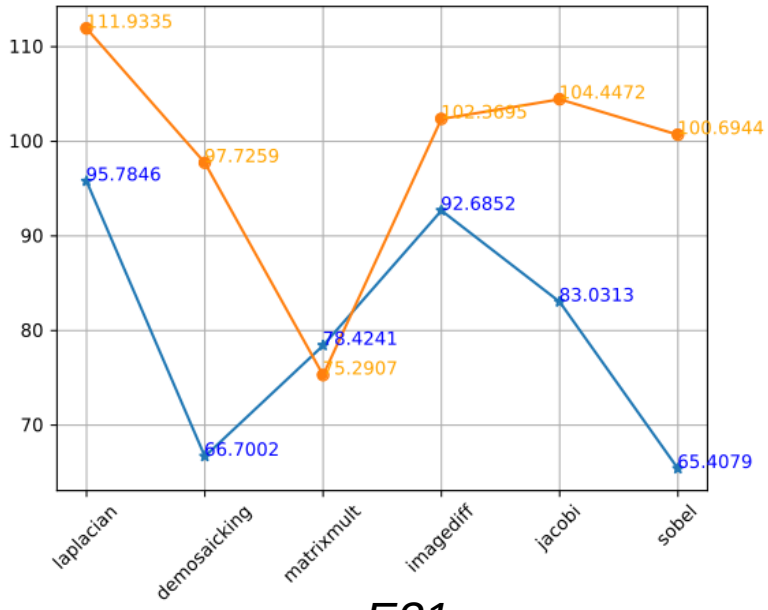
Size	Nano	Micro	Milli	Server	Fog	Campus	Facility
Example	Adafruit Trinket	Particle.io Boron	Array of Things	Linux Box	Co-located Blades	1000-node cluster	Datacenter
Memory	0.5K	256K	8GB	32GB	256G	32TB	16PB
Network	BLE	WiFi/LTE	WiFi/LTE	1 GigE	10GigE	40GigE	N*100GigE
Cost	$5	$30	$600	$3K	$50K	$2M	$1000M

Count = 10^9
Size = 10^1

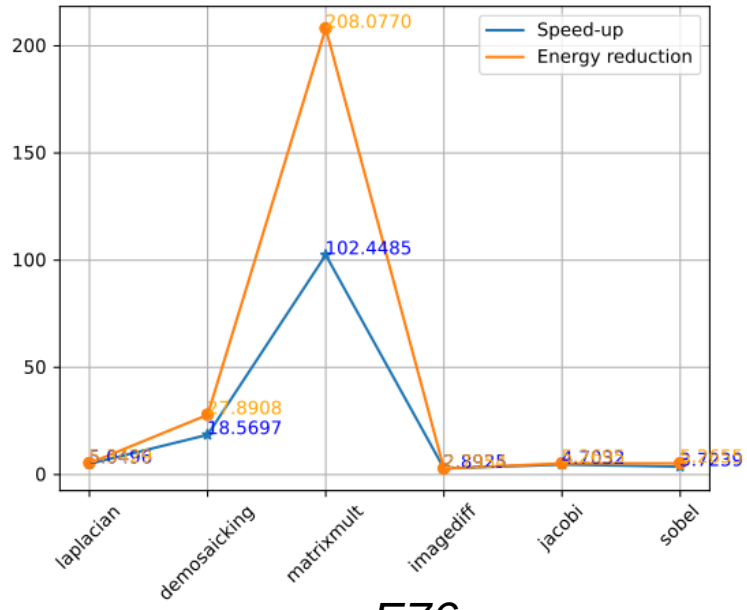
Count = 10^1
Size = 10^9

Expérience ## – Speed-up / réduction énergétique apportée par le DMU

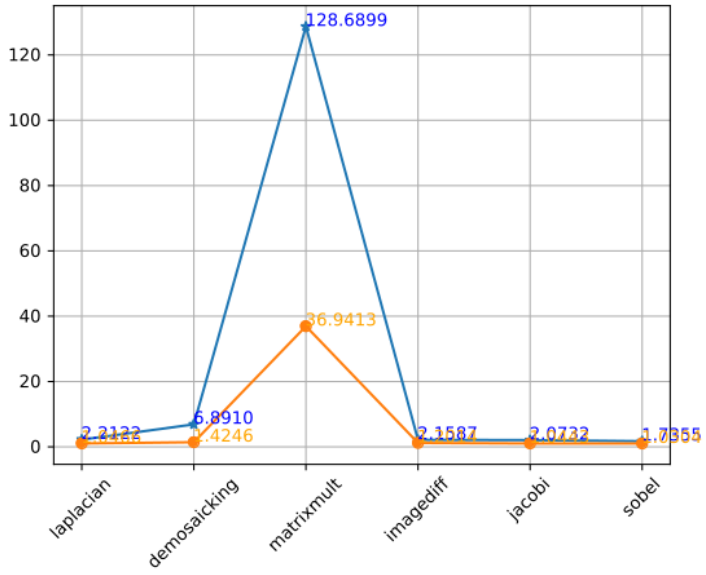
Jeu de données : 1MB



E31



E76

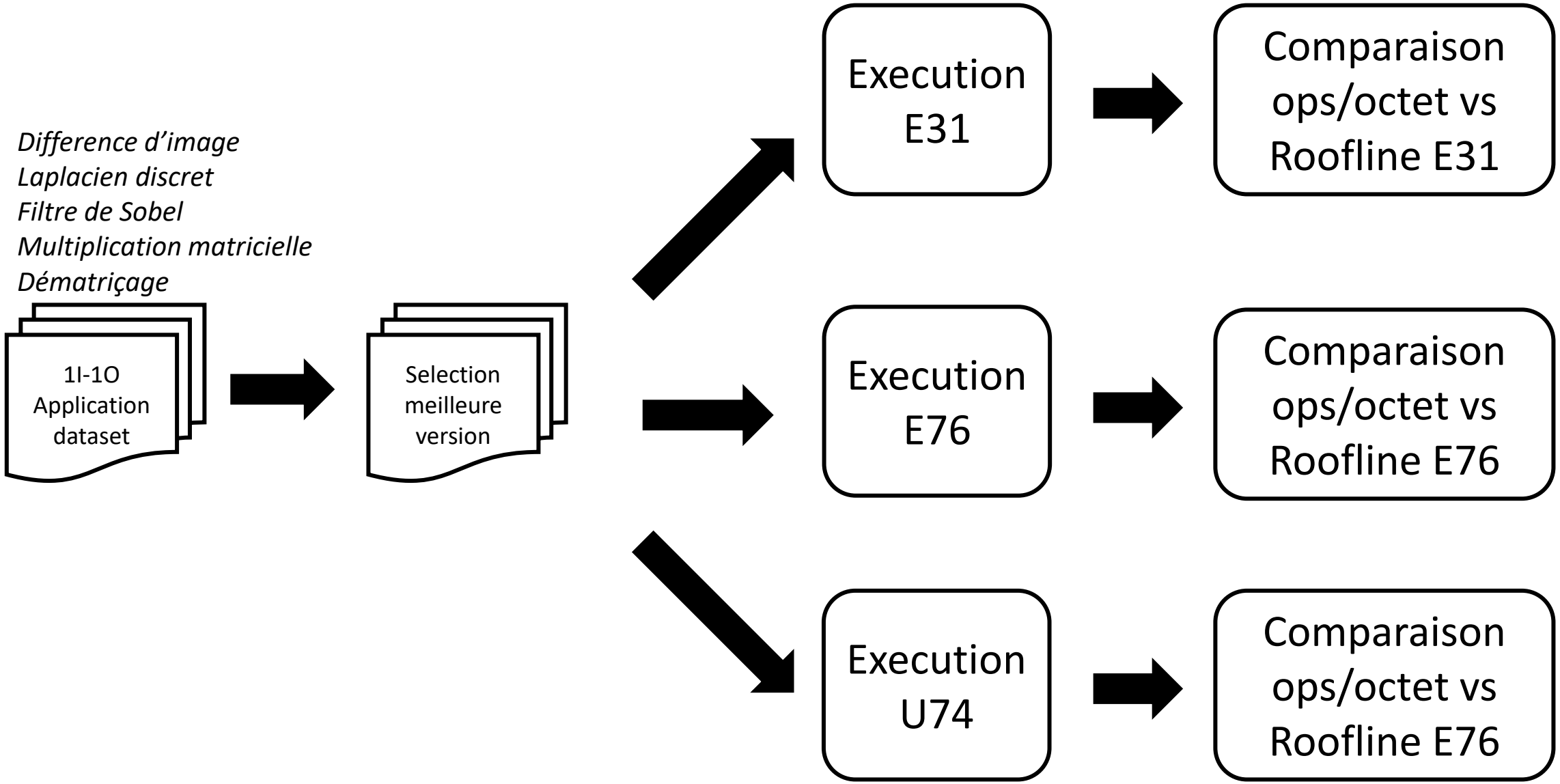


U74

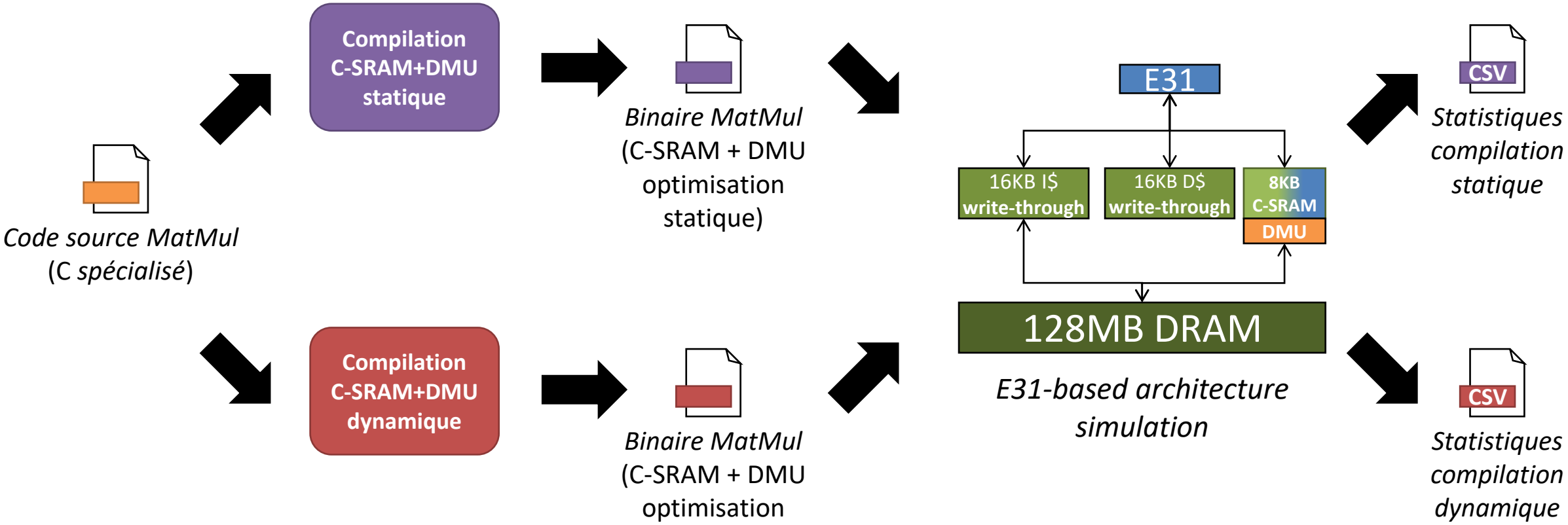
Architecture		E31	E76	U74
Speed-up	***Min***	95.7846	2.8925	1.7355
	Max	65.4079	102.4485	128.6699
Energy reduction	***Min***	75.2907	2.7954	1.0304
	Max	111.9335	208.0770	36.9413

Expérience #2 – Exploitation logicielle de la C-SRAM et du DMU

- *Difference d'image*
- *Laplacien discret*
- *Filtre de Sobel*
- *Multiplication matricielle*
- *Dématriçage*



Expérience #3 – Impact de la compilation dynamique avec Hybrogen



Expérience #3 – Impact de la compilation dynamique avec Hybrogen