Portage et optimisation d'applications de traitement d'images sur architecture KALRAY MPPA-Manycore

Journées de la Compilation

Pierre GUILLOU

Centre de recherche en informatique
MINES ParisTech

4 décembre 2013

Kalray

Conception de processeurs à haute efficacité énergétique

Mppa : Multi-Purpose Processor Array

KALRAY

Conception de processeurs à haute efficacité énergétique

MPPA MANYCORE

- processeur grande efficacité énergétique
- 256 cœurs de calcul
- faible consommation énergétique (5W)
- systèmes embarqués

KALRAY

Conception de processeurs à haute efficacité énergétique

MPPA ACCESSCORE

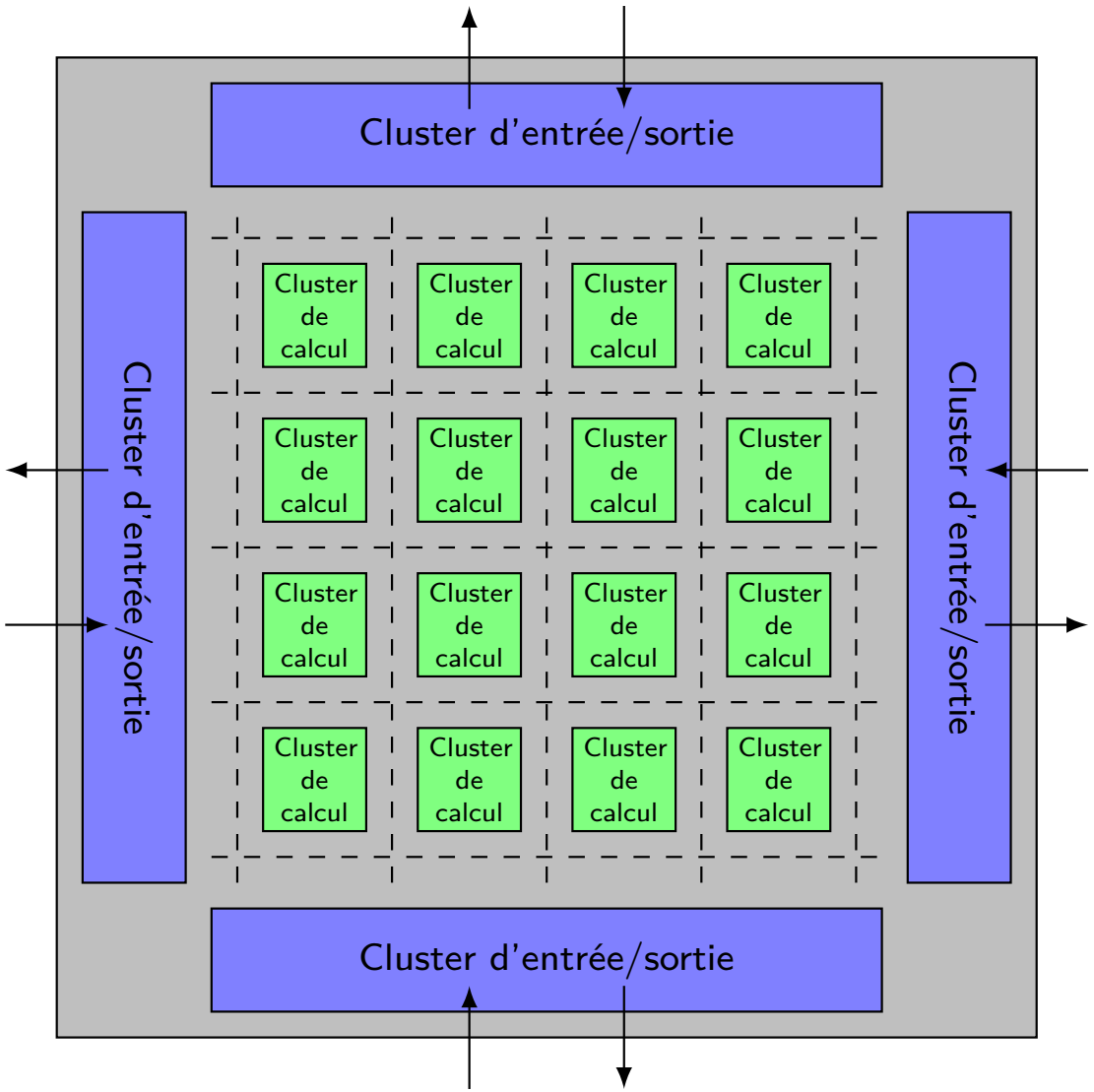
- suite logicielle
- IDE (plugin Eclipse)
- débugger
- profiler
- système de traces

KALRAY

Conception de processeurs à haute efficacité énergétique

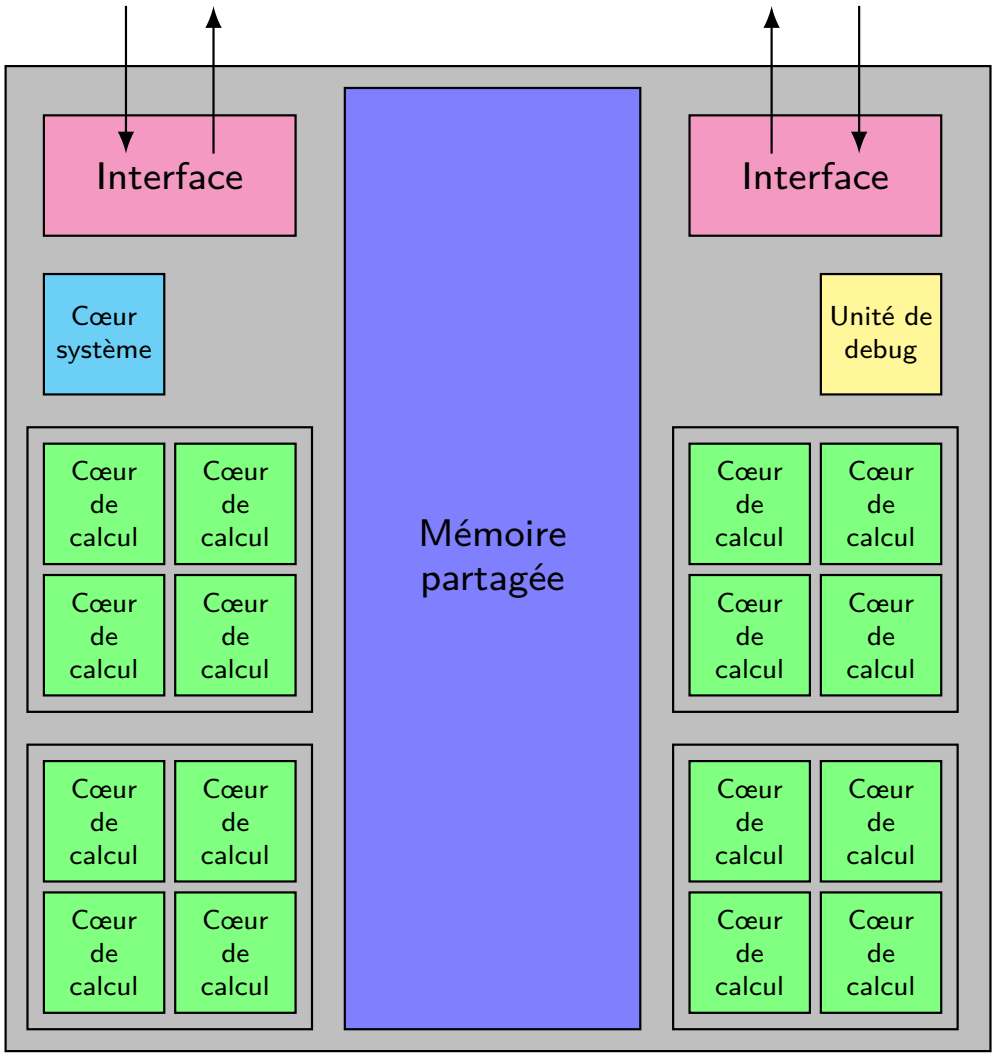
- station de travail
- plate-forme de développement complète
- MPPA-Manycore sur carte d'extension PCI-Express
- MPPA-Accesscore
- Fedora 17, 16 Go DDR

Network on Chip



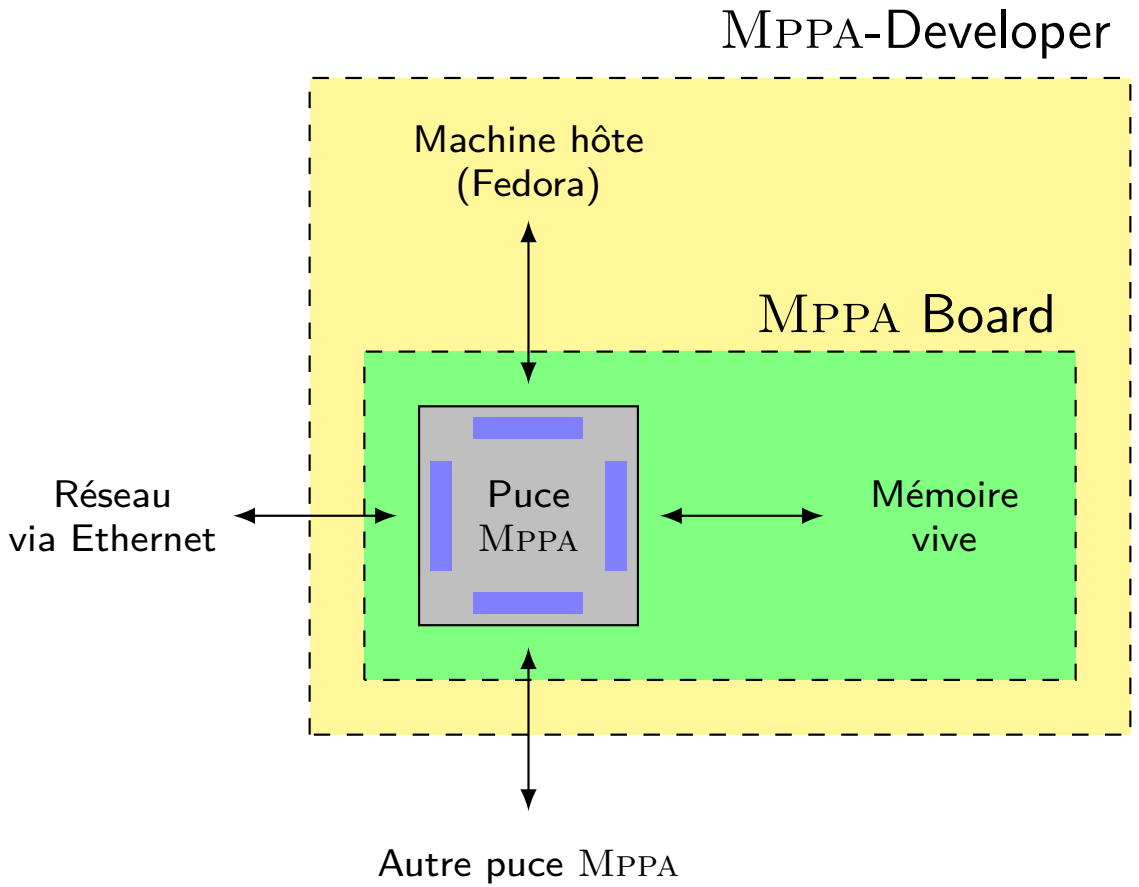
- 16 clusters de calcul
- 4 clusters d'entrée/sortie
- NoC
- gravure en 28 nm

Clusters de calcul

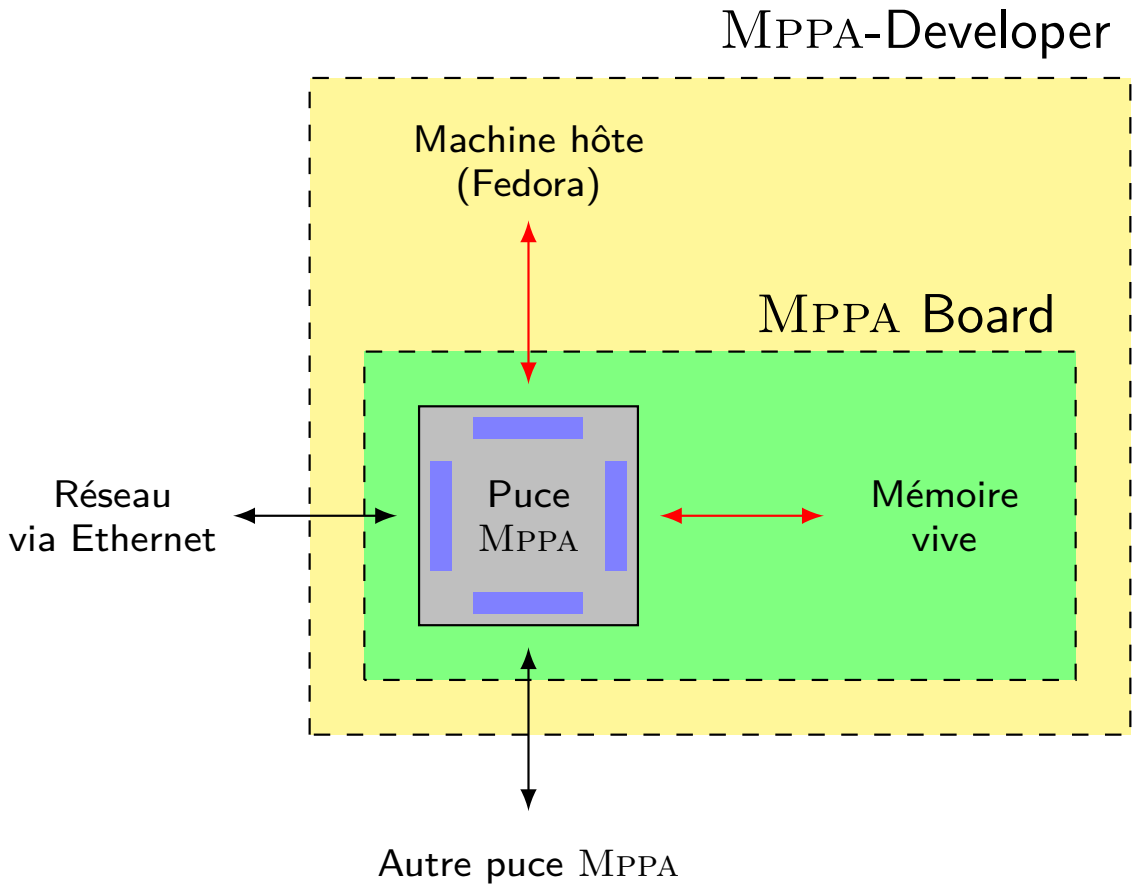


- 64 registres 32 bits
- VLIW + vectoriel
- Mémoire partagée : 2 Mo/cluster
- NodeOS

Entrées/sorties



Entrées/sorties



Deux paradigmes

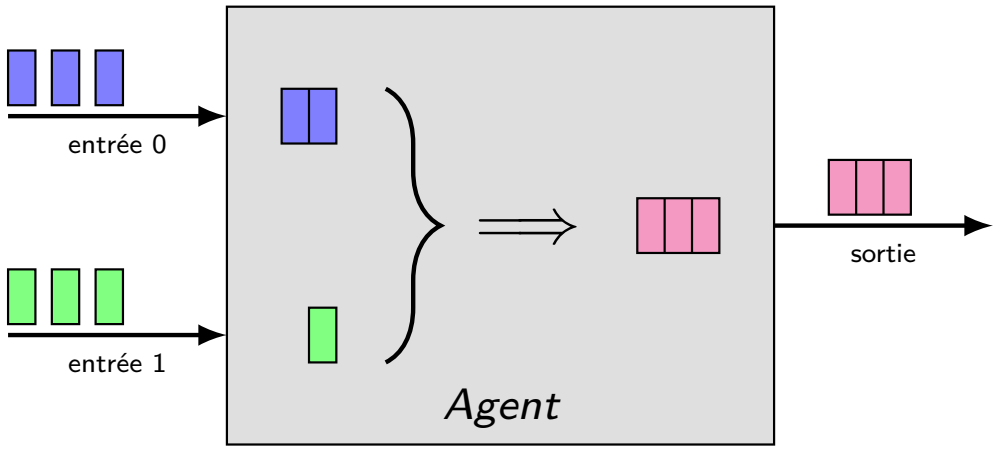
Explicitement parallèle (bas niveau)

- POSIX-Threads
- OpenMP
- MPI (portage en cours)
- OpenCL (portage en cours)

Dataflow (haut niveau)

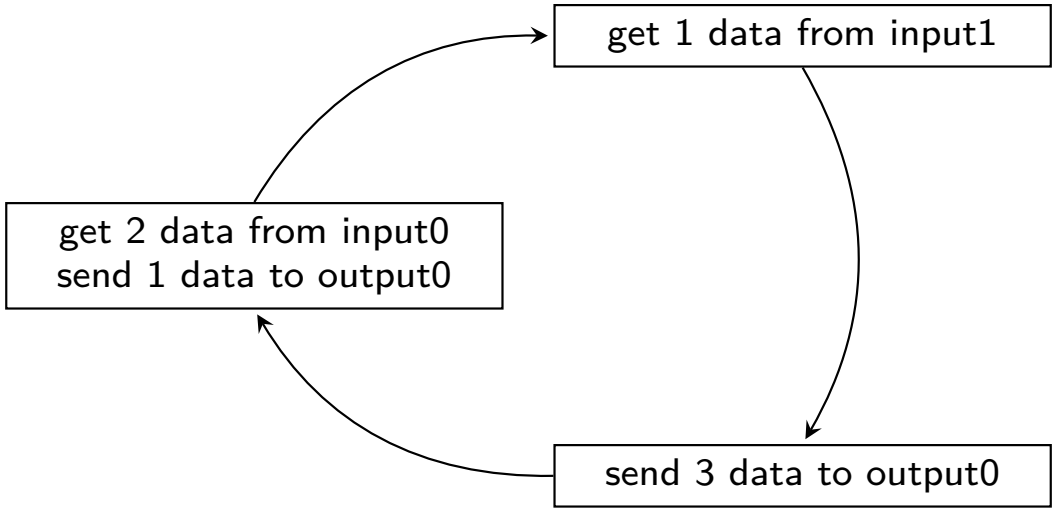
- Langage *Sigma-C*

Le langage *Sigma-C* - les *agents*



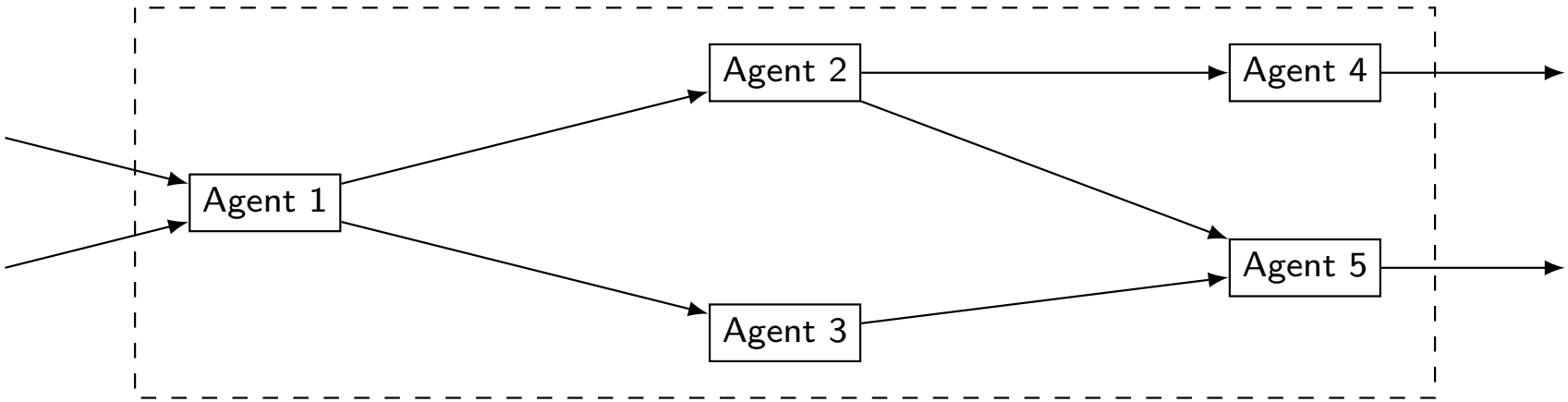

```
agent ag_name (parameters) {
  interface {
    in<input_t> input0;
    in<input_t> input1;
    out<output_t> output;
    spec { {input0[2]; output0} ; {input1} ; {output0[3]} };
  }
  ...
}
```

Les *agents*, des machines à états



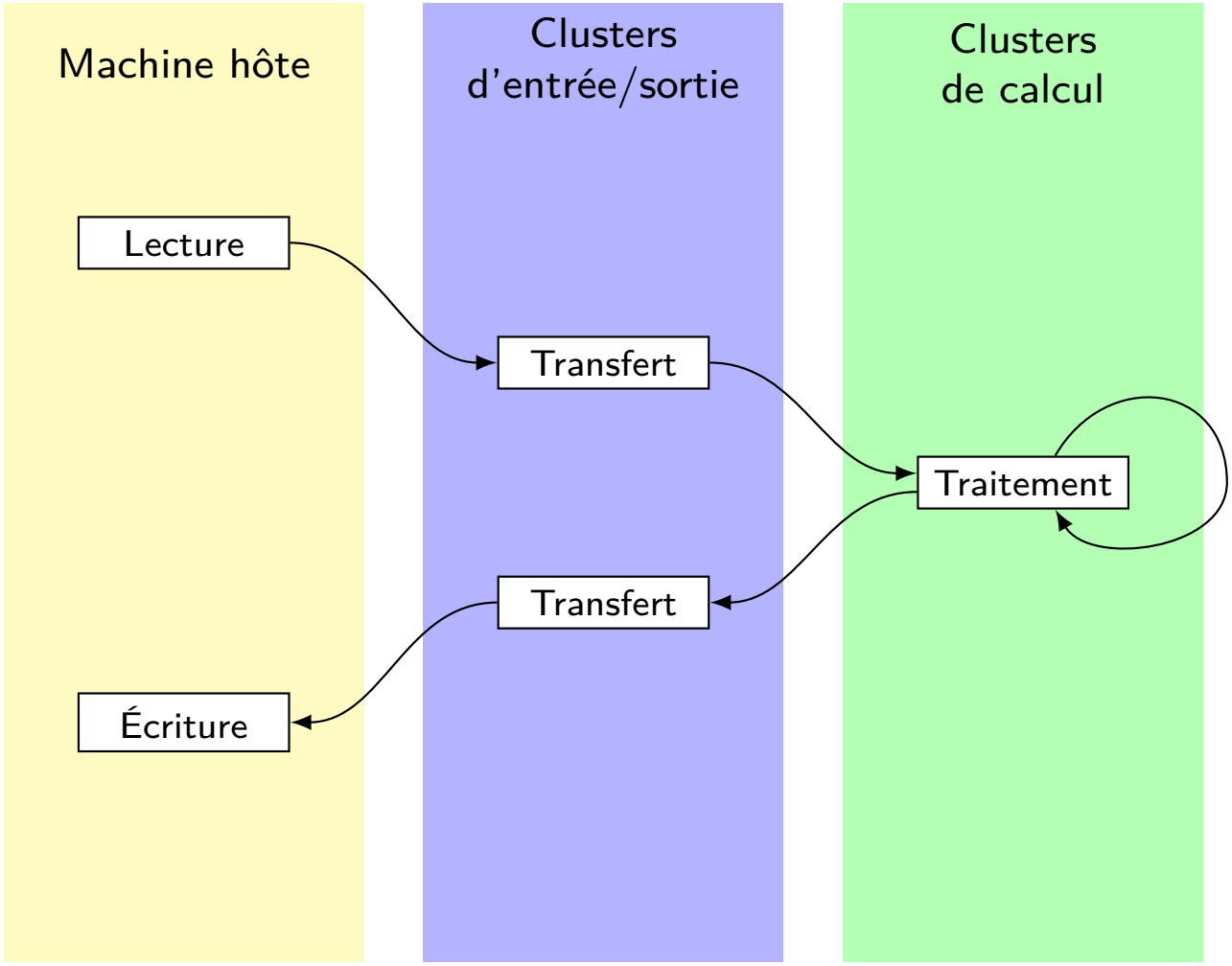

```
interface {
  ...
  spec { {input0[2]; output0} ; {input1} ; {output0[3]} };
}

void foo(int a) exchange (input0 inp[2], output0 outp) {
  outp = a*inp[0] + inp[1];
}
...
void start() {
  foo(2);
  ...
}
```

Le langage *Sigma-C* - les *subgraphs*




```
subgraph subgr () {
  ...
  map {
    agent ag1 = new agname (...);
    agent ag2 = new agname (...);
    ...
    connect (ag1.output0, ag2.input);
    connect (ag1.output1, ag3.input);
    ...
  }
}
```

Accélération matérielle sur MPPA-Manycore

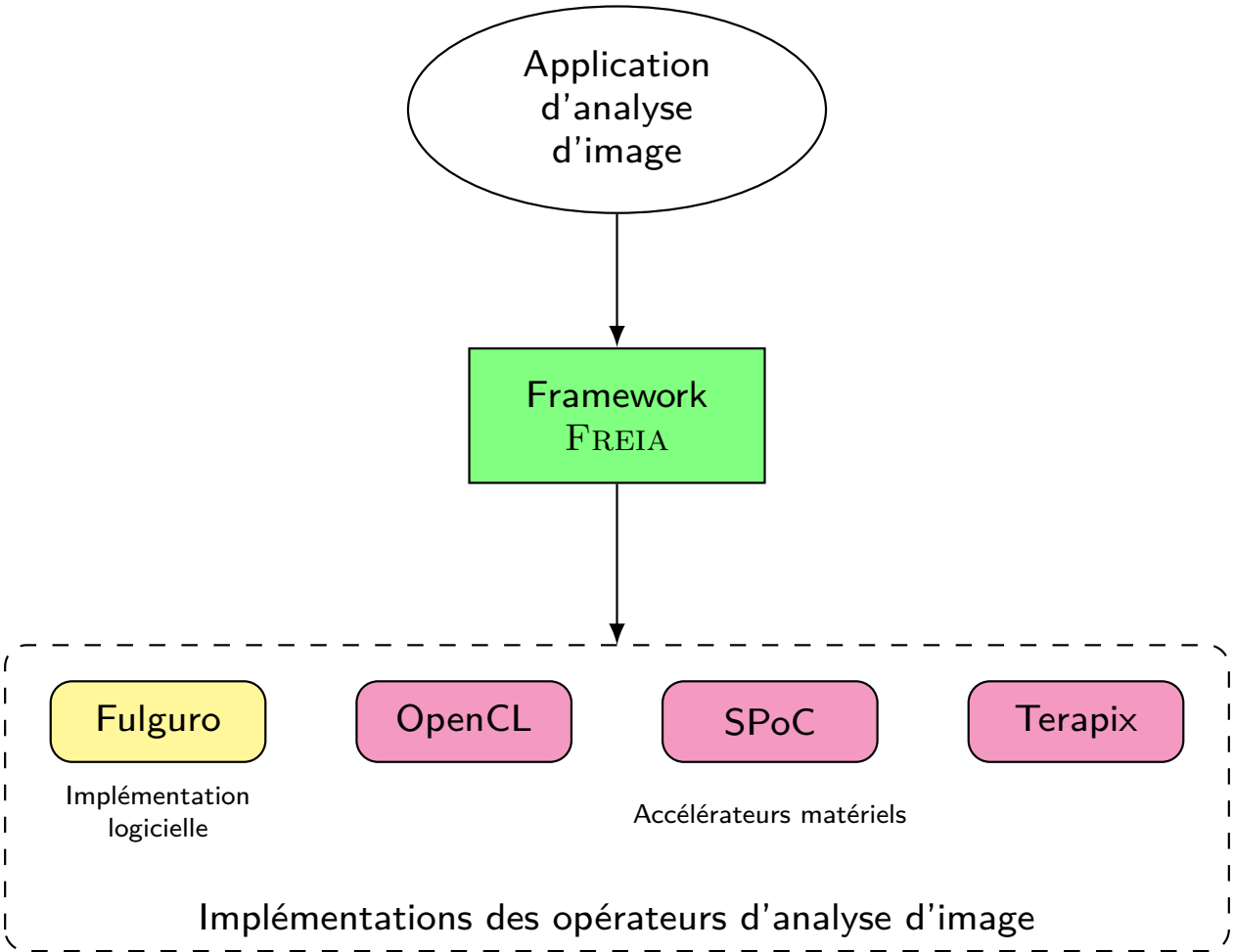


Développement d'agents *Sigma-C* pour l'analyse d'images

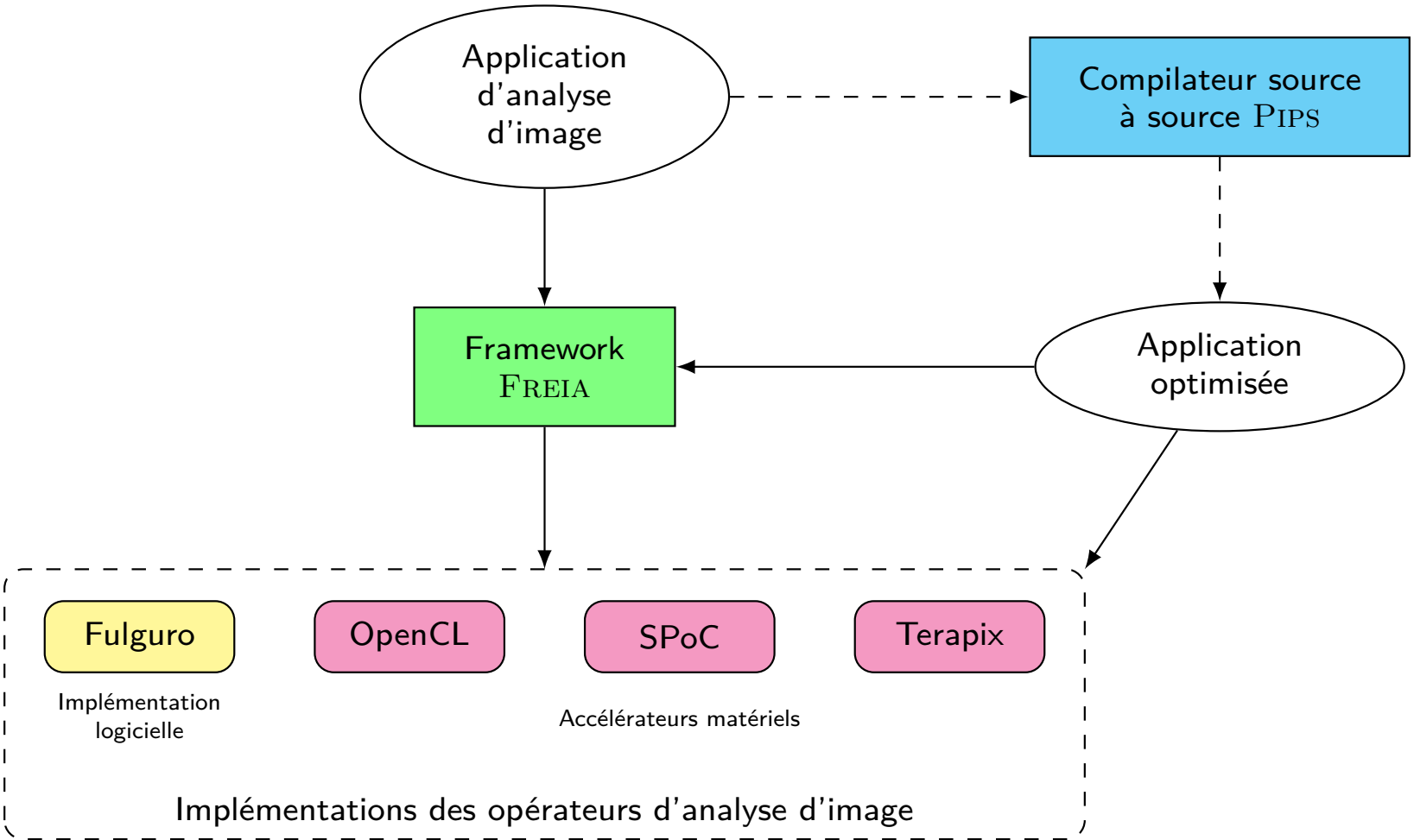
Plusieurs types d'agents développés

- opérateurs arithmétiques
  - unaires (une image en entrée + un paramètre)
  - binaires (deux images en entrée)
- opérateurs morphologiques
  - érosion, dilation, convolution
  - cœur de boucle en assembleur (développé par KALRAY)
- opérateurs de mesure
  - min et max (avec ou sans leurs coordonnées), volume

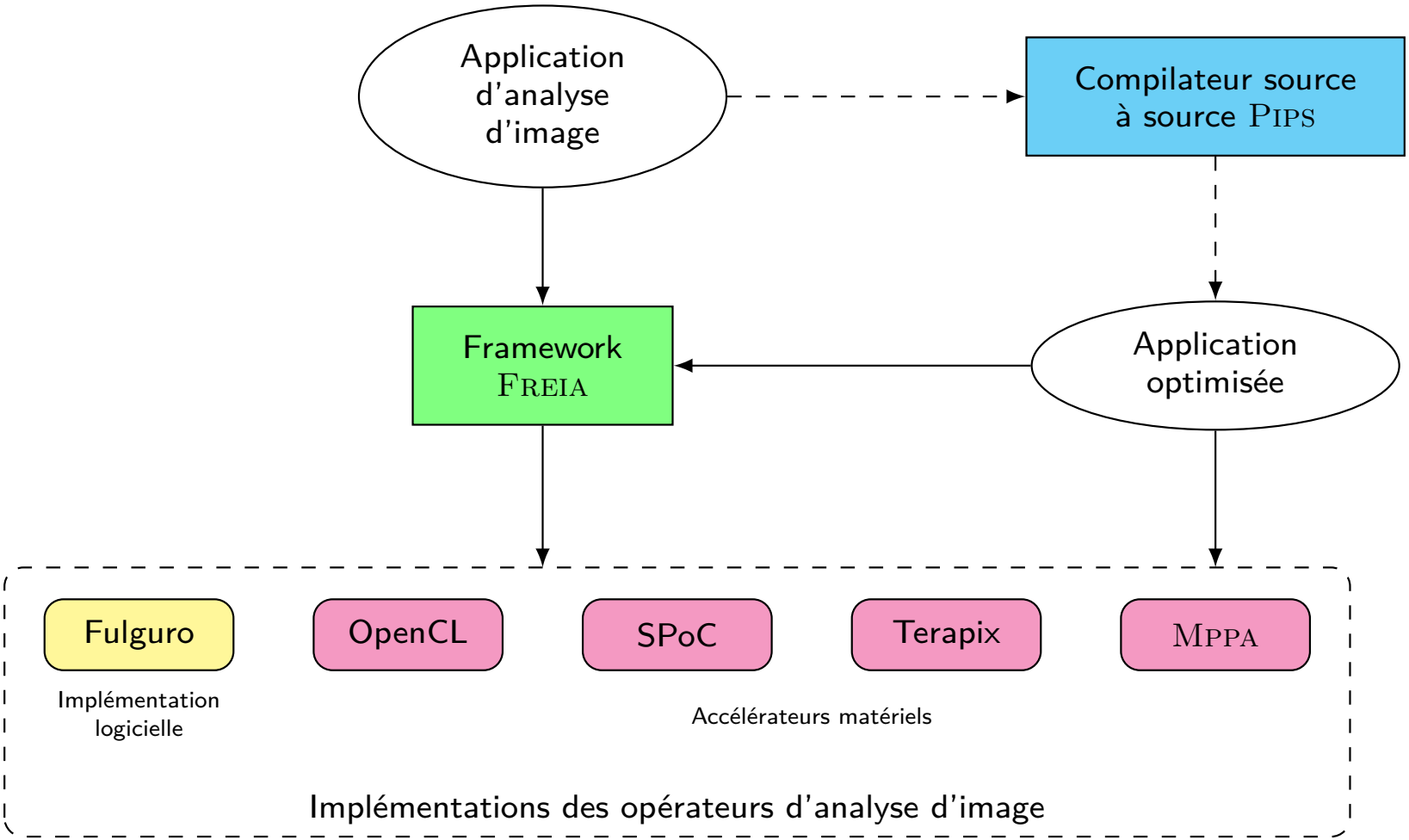
FREIA, un framework pour l'analyse d'image



FREIA, un framework pour l'analyse d'image



FREIA, un framework pour l'analyse d'image



Principe

Passer de

```
freia_aipo_erode_8c(imerode, imin, kernel);
freia_aipo_dilate_8c(imdilate, imerode, kernel);
freia_aipo_and(imout, imdilate, imin);
```

à

```
subgraph foo() {
  ...
  int16_t kernel[9] = {0,1,0, 0,1,0, 0,1,0};
  ...
  agent ero = new img_erode(kernel);
  agent dil = new img_dilate(kernel);
  agent and = new img_and_img();
  ...
  connect(ero.output, dil.input);
  connect(dil.output, and.input);
  ...
}
```

Compilation de code *Sigma-C*

Structures intermédiaires

DAG liste des sommets, entrées et sorties

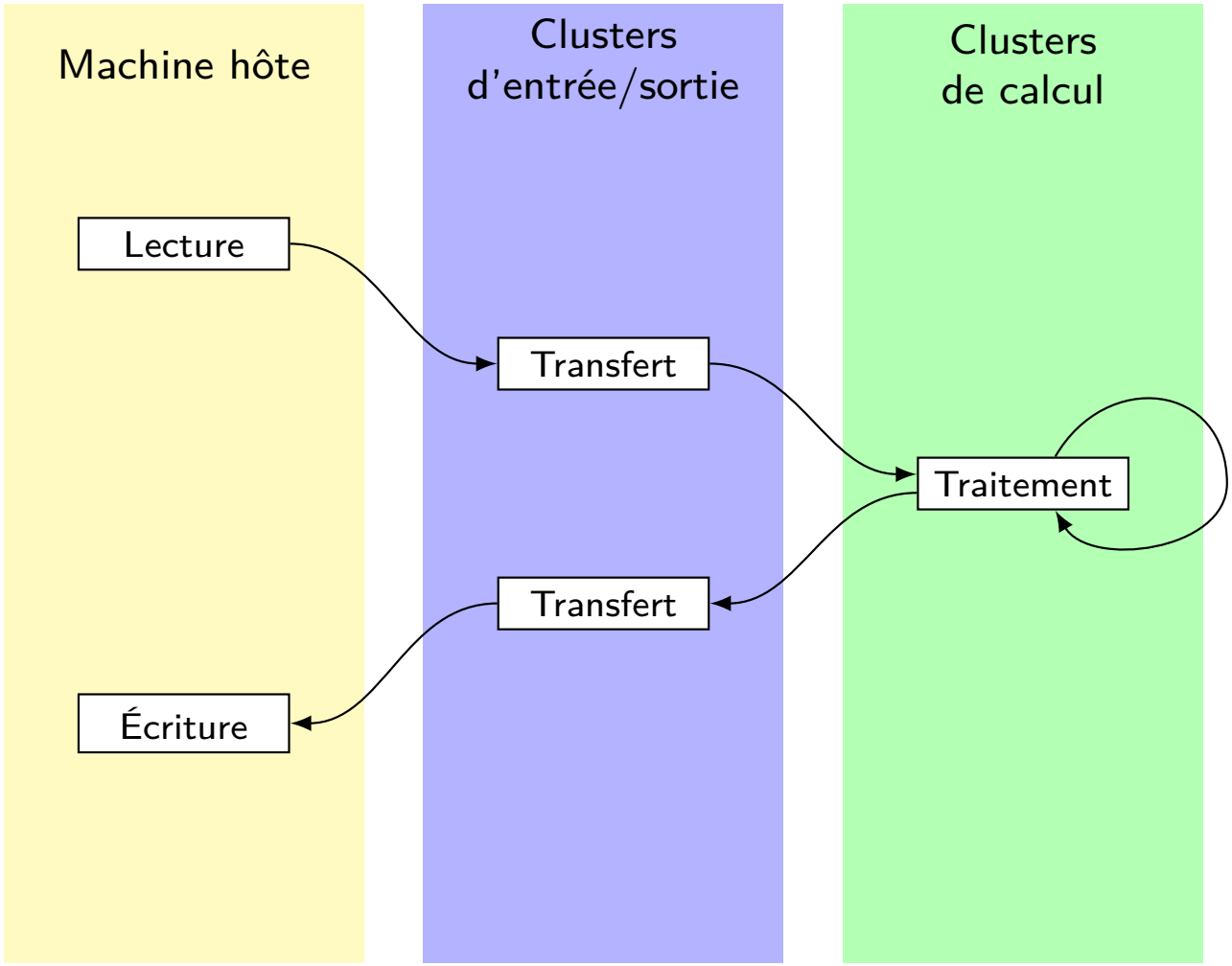
vertex opération, paramètres, liste des successeurs

Génération de code *Sigma-C*

Pour chaque vertex :

- déclarer les paramètres
- instancier l'agent correspondant avec les bons paramètres
- connecter l'agent à ses successeurs
  - insérer des agents de réplication si plus de sorties que spécifié

Retour sur le modèle d'accélérateur matériel



Inconvénients du modèle

Transferts inter clusters

- communications : 100 cycles
- mémoire d'un cluster : 2 Mo

$\Longrightarrow$ transferts ligne à ligne

Transferts hôte - clusters

Transferts + coûteux

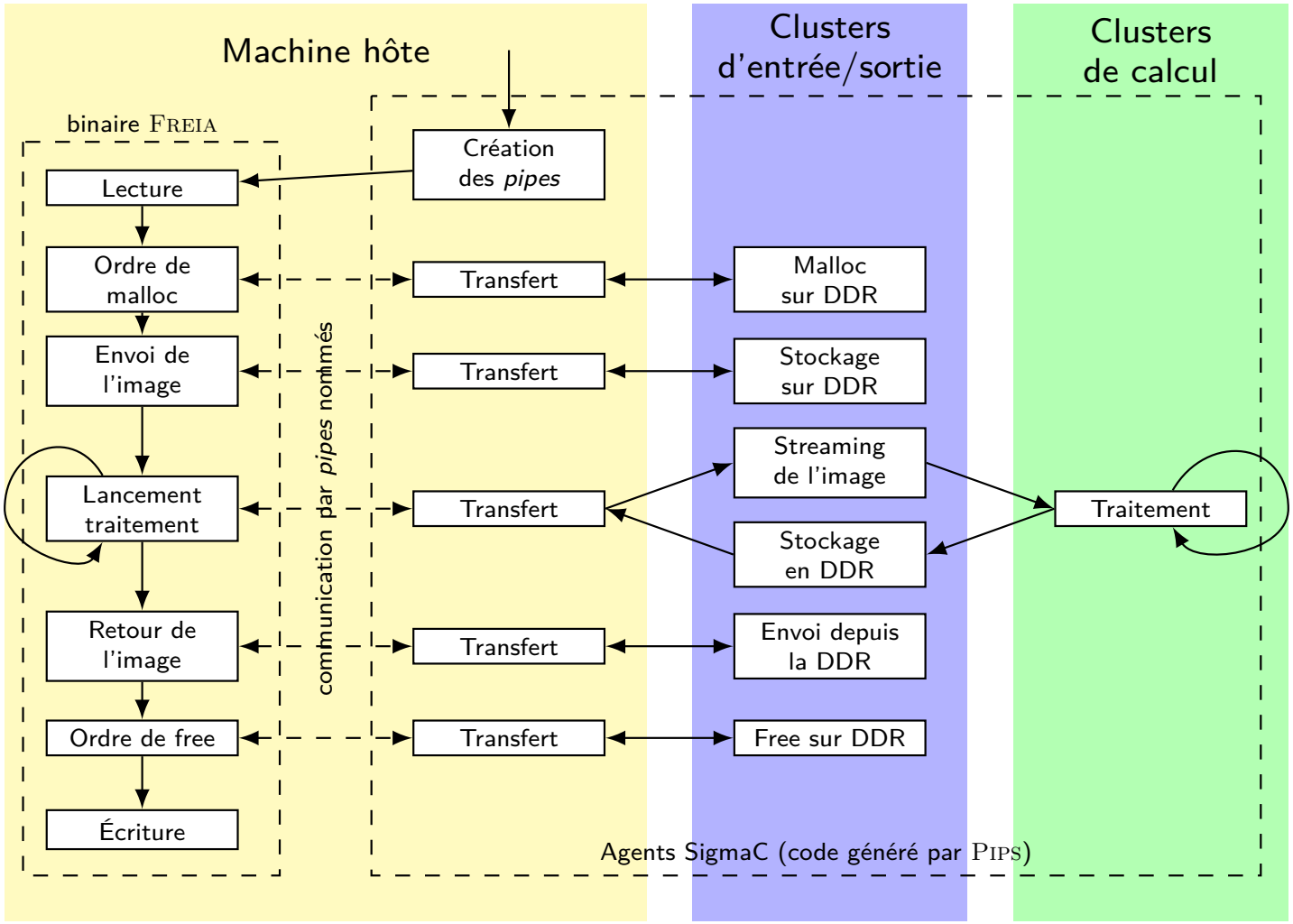
- transfert d'un seul tenant dans la mémoire vive de la puce
- streaming ligne par ligne de la mémoire vers les clusters de calcul

Inconvénients du modèle

Lecture et écriture sur disque

- garder la dépendance à FREIA
  - chargement et sauvegarde des images
- architecture en deux exécutables
  - FREIA manipule les images
  - *Sigma-C* effectue les traitements
- communications par *pipes* nommés (design de KALRAY)

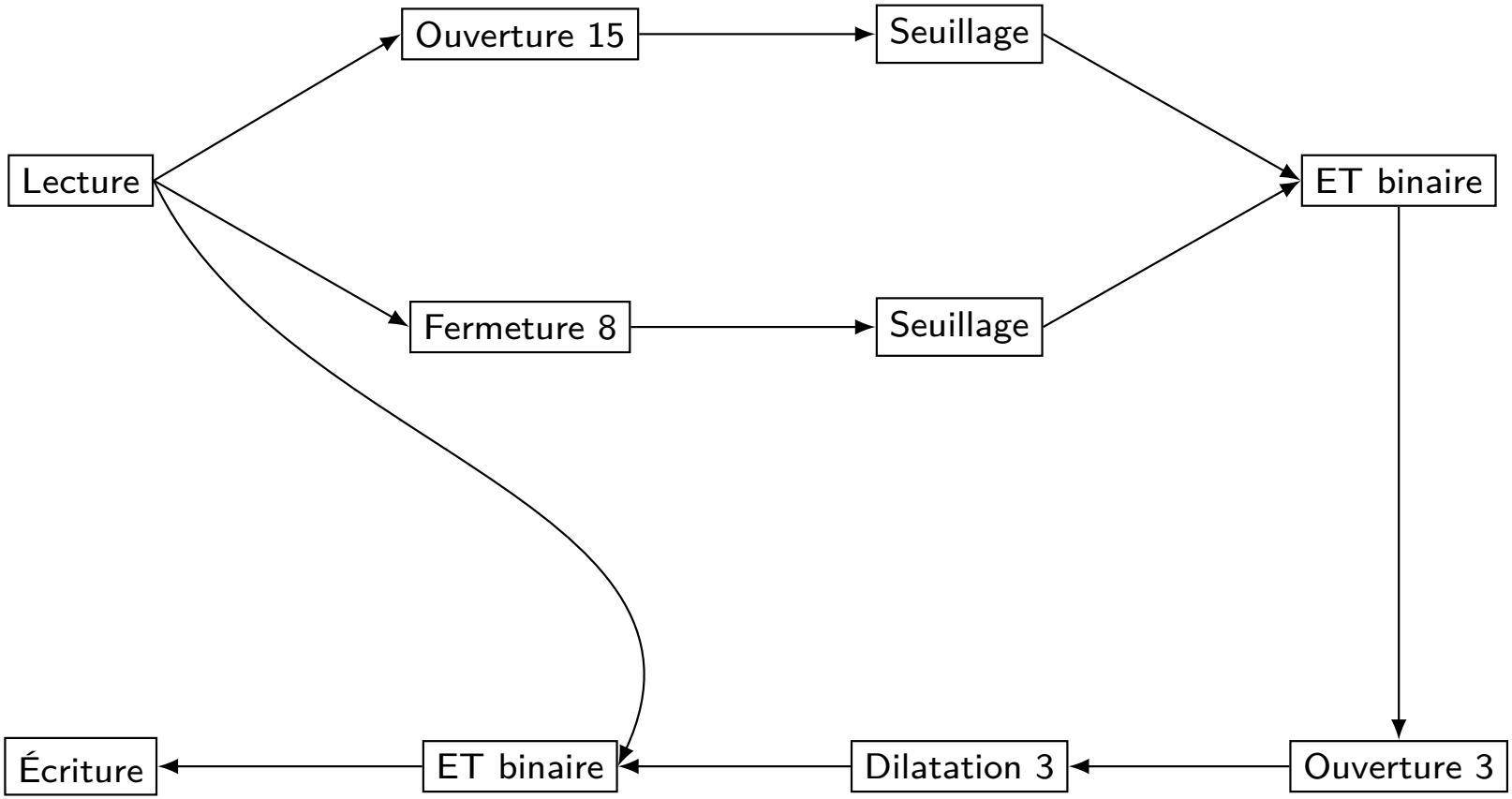
Architecture d'une application *Sigma-C* - FREIA



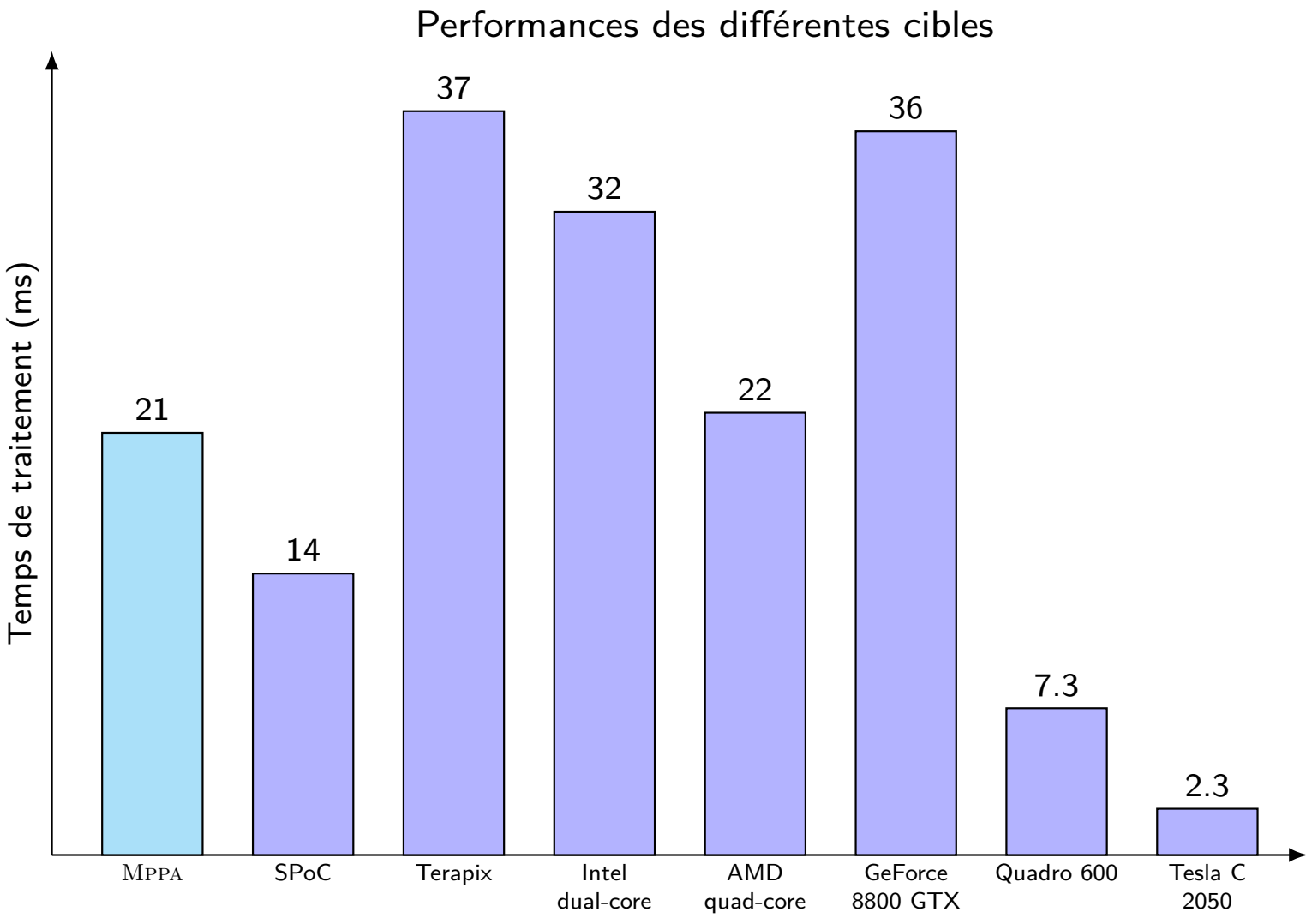
Application licencePlate - présentation

Application licencePlate - présentation

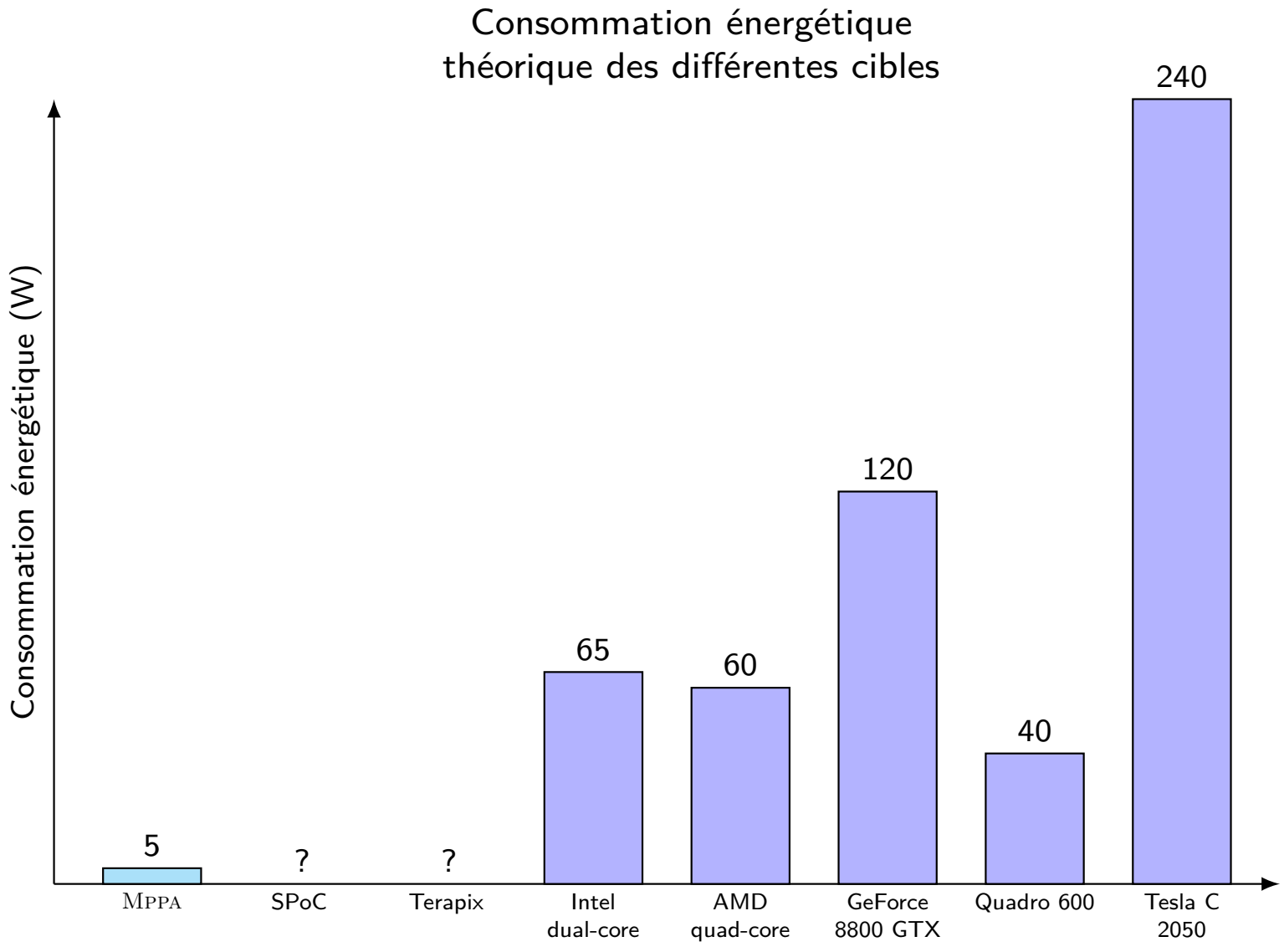
Application licensePlate - DAG



Comparaison des temps de traitement



Consommation énergétique



Résultats obtenus sur les autres applications

App.	MPPA	SPoC	Terapix	Intel dual-core	AMD quad-core
anr999	29	0.9	3.5	12.2	7.2
licensePlate	21	13.6	36.8	32.2	21.9
oop	35	124.6	63.3	12.3	8.3
toggle	33	12.6	4.3	15.0	9.0

App.	MPPA	GeForce 8800 GTX	Quadro 600	Tesla C 2050
anr999	29	9.8	1.5	0.9
licensePlate	21	36.6	7.3	2.3
oop	35	5.8	1.8	1.0
toggle	33	6.3	1.4	0.7

En conclusion

- bibliothèque d'agents *Sigma-C* pour l'analyse d'image
- génération via PIPS de code applicatif intégré au framework FREIA
- plusieurs applications fonctionnelles
- optimisations des temps d'exécution
- future work :
  - amélioration performances,
  - support d'autres applications
  - ...

Portage et optimisation d'applications de traitement d'images sur architecture KALRAY MPPA-Manycore

Journées de la Compilation

Pierre GUILLOU

Centre de recherche en informatique
MINES ParisTech

4 décembre 2013