

# Programming the FlexRAM Parallel Intelligent Memory System



B. B. Fraguela\*, J. Renau†, P. Feautrier‡  
D. Padua† and J. Torrellas†

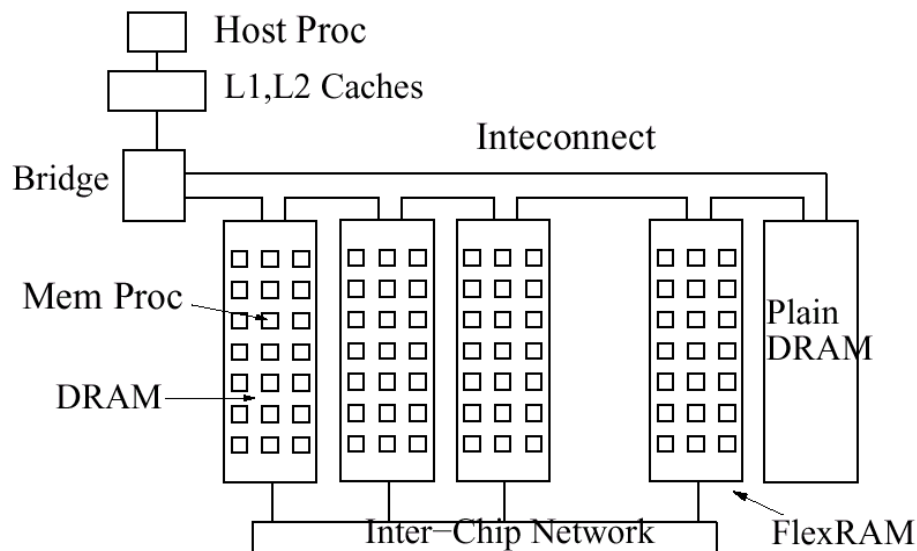
\*Univ. da Coruña

†Univ. of Illinois

‡ENS de Lyon

# Intelligent Memory Architectures

⌘ Main memory enhanced with many simple processors



- Heterogeneous
- Highly parallel

Problem: Little research on how to program these architectures

# Contributions



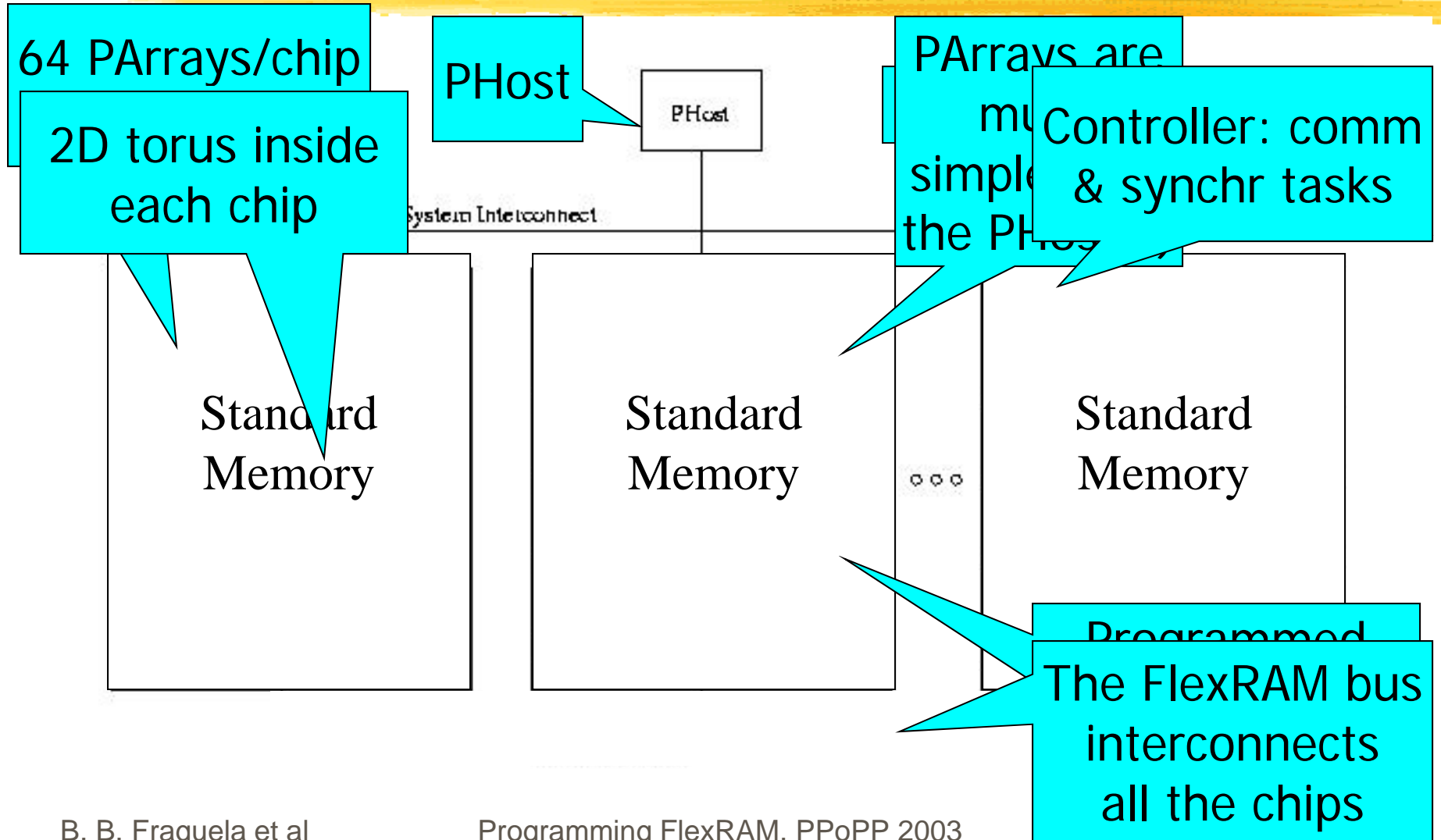
- ⌘ Language support for intelligent memories
  - ☑ OpenMP-like directives (CFlex)
  - ☑ Library of Intelligent Memory Operations (IMOs)
- ⌘ Runtime system and OS extensions
- ⌘ Speedups of over one order of magnitude

# Outline



- ⌘ FlexRAM Architecture
- ⌘ Software Support
- ⌘ CFlex
- ⌘ IMO
- ⌘ Evaluation
- ⌘ Related Work
- ⌘ Conclusions

# FlexRAM Architecture



# FlexRAM Architectural Issues



- ⌘ PArrays cannot interrupt/invoke the PHost(s)
  - ☑ PArray requests are sent to chip controller
  - ☑ PHost polls the controllers and services the requests
- ⌘ Communication PHost - PArray : pass input and output arguments through memory
- ⌘ No HW cache coherence
  - ☑ Compiler inserted cache flushes and invalidations

# Outline



- ⌘ FlexRAM Architecture
- ⌘ Software Support
- ⌘ CFlex
- ⌘ IMO
- ⌘ Evaluation
- ⌘ Related Work
- ⌘ Conclusions

# Operating System Extensions



- ⌘ Common address space for PHost(s) and PArrays
- ⌘ PArrays kernel
  - ☑ Manages the TLB
  - ☑ Manages spawn and termination of local tasks
- ⌘ PHost OS
  - ☑ Updates the shared page table
  - ☑ First-touch placement of pages
  - ☑ Cooperates with PArray kernels to keep TLBs coherent



# Programming FlexRAM



- ⌘ OpenMP-like directives (CFlex)
- ⌘ Library of Intelligent Memory Operations (IMOs)

# CFlex



⌘ CFlex: family of directives inspired by OpenMP

☒ Execution modifiers: build/sync tasks

☒ Data modifiers: properties of data structures

☒ Executable directives: barriers, prefetches,...

```
#pragma FlexRAM directive-type [clauses]
```

# Execution Modifier: Spawn

⌘ *Directive-type*

⊞ [**phost** | **parray**] : specifies kind of processor to use

⌘ *Clauses*

⊞ **on\_home**(*x*) : run the task on the PArray on whose bank *x* is located.

⊞ **sync** / **async**: parent task must stop until child finishes (**sync**) or not (**async**)

⊞ **pfor**: parallelize **for** loop

# Execution Modifier: Spawn (II)

## ⌘ *clauses* (cont.):

- ☒ **if**(cond) / **else**: conditional execution of the compiler directive
- ☒ **shared**, **private**, **firstprivate**, **lastprivate**, **reduction**: scope clauses with the same meaning as in OpenMP
- ☒ **flush**: specify which pieces of data to flush from PHost cache

# Example: Parallelizing a Loop



```
for(p = head; p != NULL; p = p->next)
    process(p->data);
```

# Example: Parallelizing a Loop



```
for(p = head; p != NULL; p = p->next)
    process(p->data);
```

# Example: Parallelizing a Loop

```
#pragma FlexRAM phost sync
    for(p = head; p != NULL; p = p->next)
#pragma FlexRAM parray async on_home(*(p->data)) \
                                firstprivate(p)
        process(p->data);
```

# Example: Parallelizing a Loop



```
#pragma FlexRAM parray pfor on_home(*(p->data)) \  
                                firstprivate(p)  
for(p = head; p != NULL; p = p->next)  
    process(p->data);
```



# Parallelizing Complex Codes



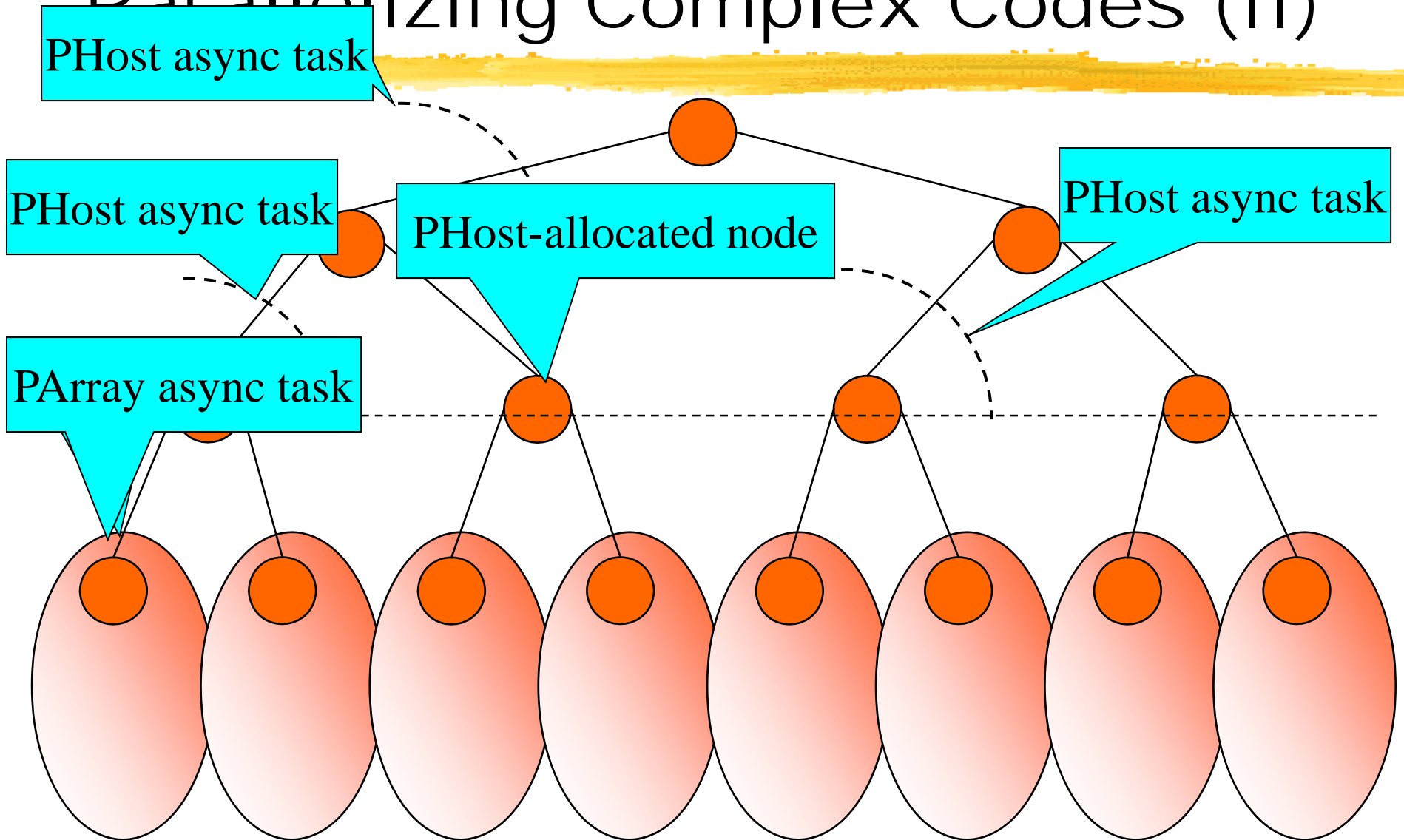
```
int TreeAdd (register tree_t *t) {
    if (t == NULL) return 0;
    else {
        int leftval, rightval;

        leftval = TreeAdd(t->left);

        rightval = TreeAdd(t->right);

        return leftval + rightval + t->val;
    }
}
```

# Parallelizing Complex Codes (II)



# Parallelizing Complex Codes (III)

```
int TreeAdd (register tree_t *t) {
    if (t == NULL) return 0;
    else {
        int leftval, rightval;

        #pragma FlexRAM phost sync
        {
            #pragma FlexRAM parray async on_home(*(t->tleft)) if (lcl(t->left))
            #pragma FlexRAM phost async else
                leftval = TreeAdd(t->left);

            #pragma FlexRAM parray async on_home(*(t->tright)) if (lcl(t->right))
                rightval = TreeAdd(t->right);
        }

        return leftval + rightval + t->val;
    }
}
```

# Outline



- ⌘ FlexRAM Architecture
- ⌘ Software Support
- ⌘ CFlex
- ⌘ IMOs
- ⌘ Evaluation
- ⌘ Related Work
- ⌘ Conclusions

# Intelligent Memory Operations (IMOs)



- ⌘ Libraries that hide FlexRAM while providing near-optimal performance
- ⌘ Implement common operations on data structures often used in programs
- ⌘ Highly optimized, both the sequential and the parallel versions

# Example IMO: Vector Container

IMO description	Syntax
Apply func f with arg a	<code>Vector_apply(v, f, a)</code>
Search element that fulfills cond f with arg a	<code>Vector_search(v, f, a)</code>
Generate vector with the result of appl func f with arg a	<code>v2=Vector_map(v, f, a)</code>
Reduce vector applying func f, whose neutrum is a	<code>Vector_reduce(v, f, a)</code>
Process two vectors and an arg a, generating a new vector	<code>v3=Vector_map2(v, v2, f, a)</code>

# Outline



- ⌘ FlexRAM Architecture
- ⌘ Software Support
- ⌘ CFlex
- ⌘ IMO
- ⌘ Evaluation
- ⌘ Related Work
- ⌘ Conclusions

# Architecture Parameters



## ⌘ PHost

- ☒ 1.6 GHz, 5 issue
- ☒ L1 cache: 32 KB
- ☒ L2 cache: 1 MB

⌘ Mem latency 180 cycles

## ⌘ PArray

- ☒ 1.2 GHz, 2 issue in order
- ☒ L1 cache: 8 KB
- ☒ No FP support

⌘ Mem latency 14 cycles



# Applications (I)

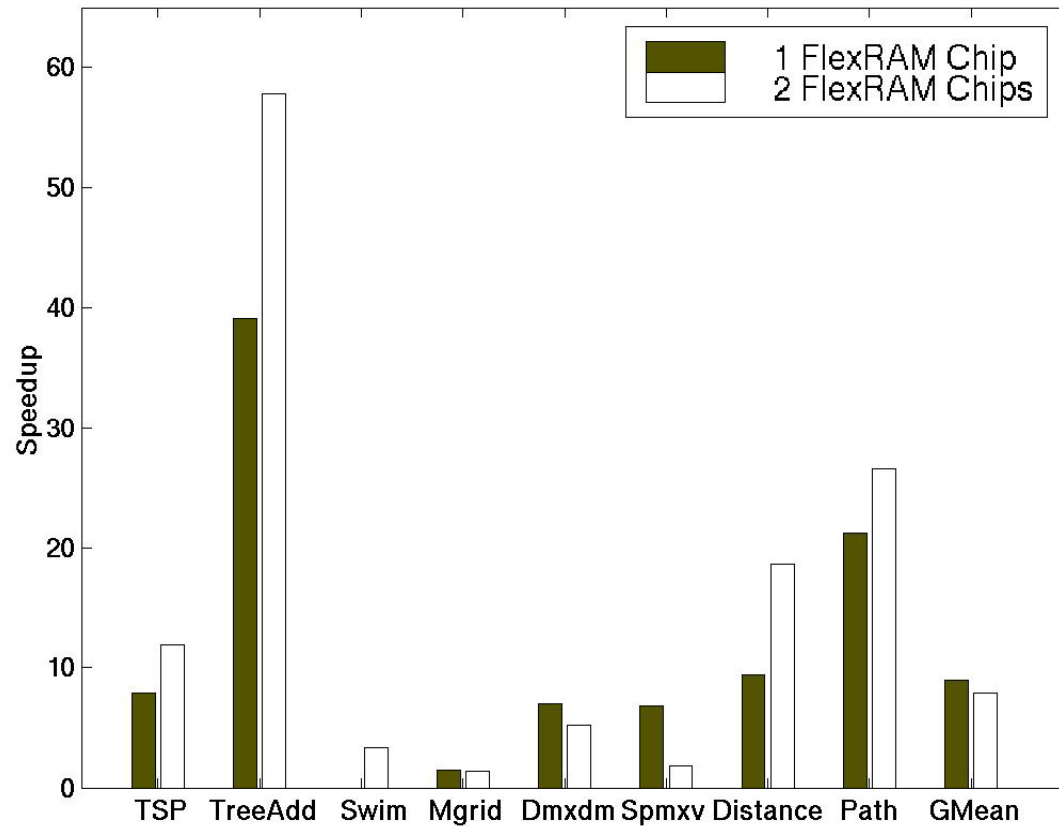
Application	Suite	Access	Data	Task Size
TSP	Olden	Ptr	FP	Large
TreeAdd	Olden	Ptr	Int	Large
Swim	SPEC OMP 2001	Reg	FP	Med
Mgrid	SPEC OMP 2001	Reg	FP	Var
Dmxdm	Kernel	Reg	FP	Large
Spmxv	Kernel	Ind	FP	Med
Distance	CAM	Ptr	Int	Large
Path	CAM	Ptr	Int	Small

# Original code not changed

## Applications (II)

Application	Code size (lines)	Directives	Additional lines
TSP	485	12	5
TreeAdd	71	8	4
Swim	272	8	0
Mgrid	470	13	0
Dmxdm	81	1	2
Spmxv	47	1	0
Distance	108	17	7
Path	165	17	9

# Speedups



Speedups: Over one order of magnitude!

# Further Optimizations



- ⌘ Single task for consecutive pages affected by an **on\_home** residing in the same bank
- ⌘ Consecutive rather than cyclic spawn among the FlexRAM chips
- ⌘ Limiting the usage of **on\_home** to large loops
  
- ⌘ Increase the average speedup by about 30%

# Outline



- ⌘ FlexRAM Architecture
- ⌘ Software Support
- ⌘ CFlex
- ⌘ IMO
- ⌘ Evaluation
- ⌘ Related Work
- ⌘ Conclusions

# Other Intelligent Mem Arch.



- ⌘ Active Pages [Oskin et al], DIVA [Hall et al]
  - ☒ Program the machine by hand
  - ☒ Can also be programmed with CFlex/IMOs
  - ☒ Seem more sensitive to data placement than FlexRAM
  - ☒ Further extensions of CFlex to specify alignment and placement of data structures
  - ☒ Message passing is more natural for DIVA

# Related Work



- ⌘ FlexRAM [Solihin et al]: automatic partition and mapping by a compiler
  - ☒ Only feasible for simple codes
  - ☒ Performance is limited
- ⌘ Widespread compiler directives
  - ☒ OpenMP: UMA model, no locality clauses
  - ☒ HPF: extensive alignment + replication
  - ☒ Both: Unadequate for irregular applications

# Conclusions



- ⌘ Effective programming support for Intelligent Mem
  - ☒ CFlex: family of pragmas inspired by OpenMP
  - ☒ IMOs: library of intelligent memory operations
- ⌘ CFlex parallelizes more problems than OpenMP
- ⌘ Complexity can be further hidden using IMOs
- ⌘ Speedups over one order of magnitude



# Programming the FlexRAM Parallel Intelligent Memory System



B. B. Fraguela\*, J. Renau†, P. Feautrier‡  
D. Padua† and J. Torrellas†

\*Univ. da Coruña

†Univ. of Illinois

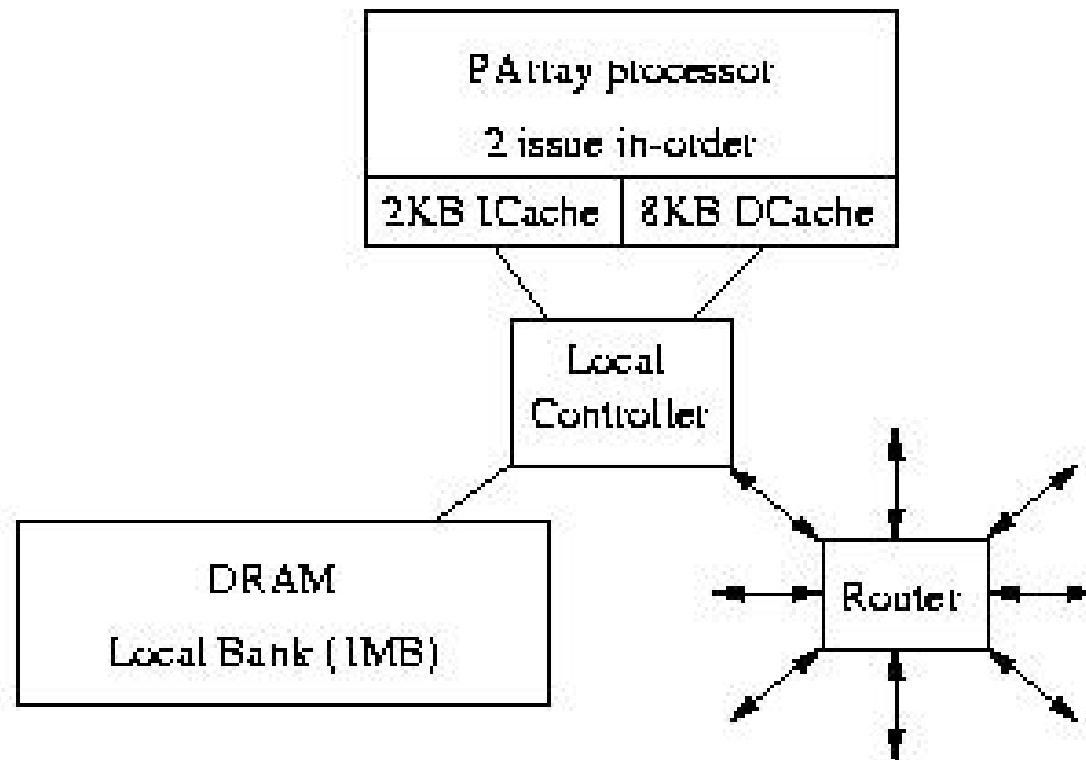
‡ENS de Lyon

# Runtime System



- ⌘ Task management
  - ☒ Creation of buffer with input args + message
  - ☒ PHost spins on a termination flag set by the PArray
- ⌘ Interface to chip controller locks and constructions built upon them, like barriers
- ⌘ Heap memory management
- ⌘ Polling of the FlexRAM chips

# PArray Structure



# Further Optimizations



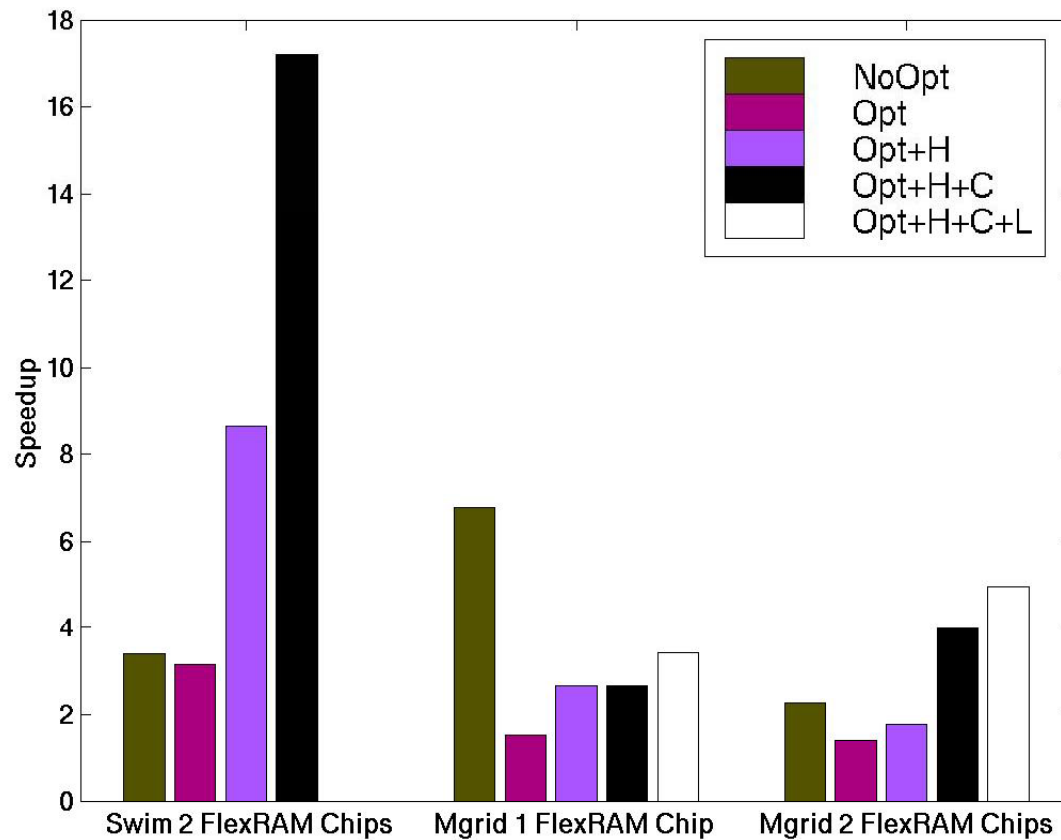
## ⌘ Initial optimizations:

- ☒ Alignments using the OS first-touch policy
- ☒ **on\_home** clause to exploit locality

## ⌘ New optimizations:

- ☒ H: single task for consecutive pages affected by an **on\_home** residing in the same bank
- ☒ C: consecutive rather than cyclic spawn among the FlexRAM chips
- ☒ L: limiting the usage of **on\_home** to large loops

# Software Optimization Results



# Final Results

