Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA
SEVERO
OCHOA

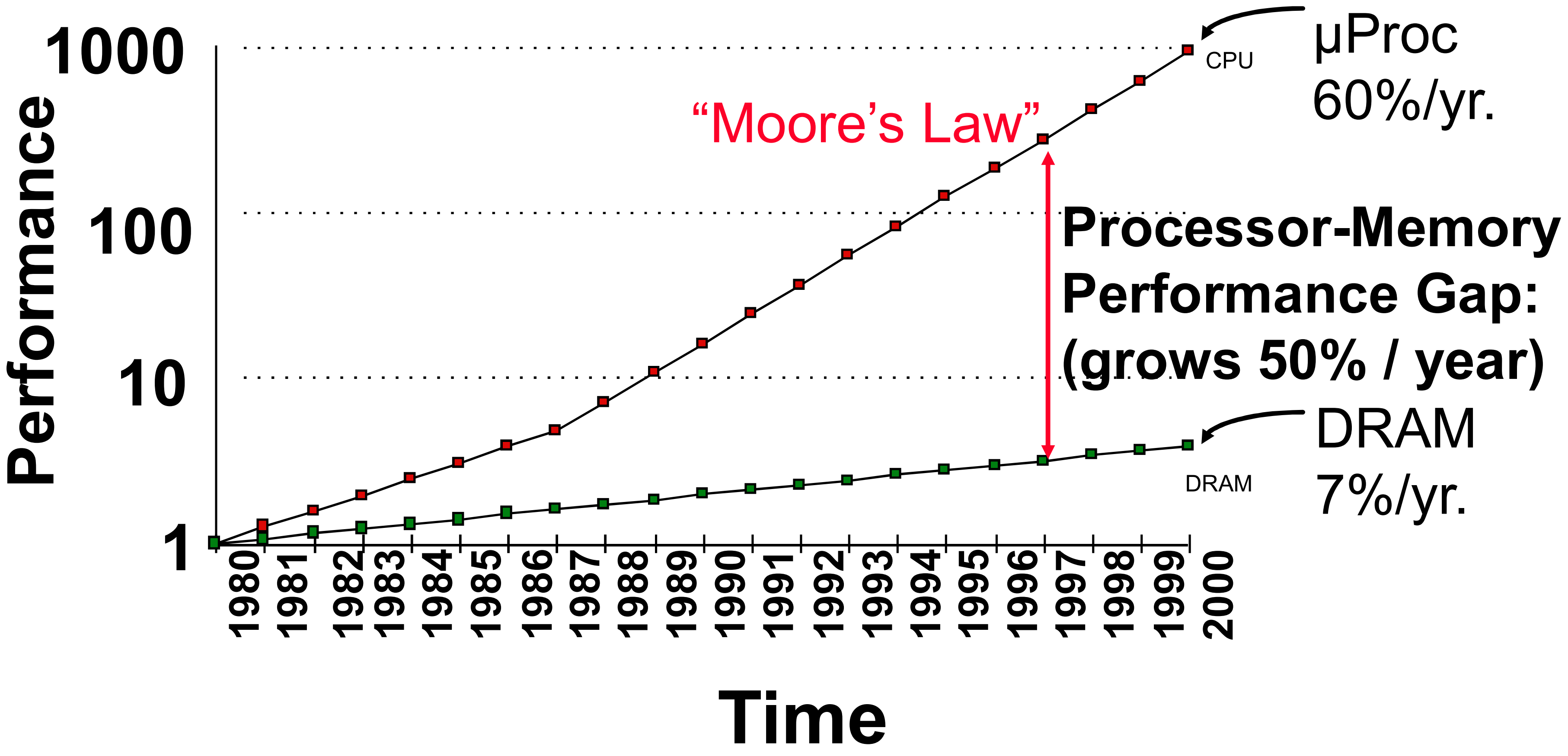
Kilo Instruction Processors

Adrián Cristal

2/7/2019

YALE 80

Processor-DRAM Gap (latency)



µProc
60%/yr.

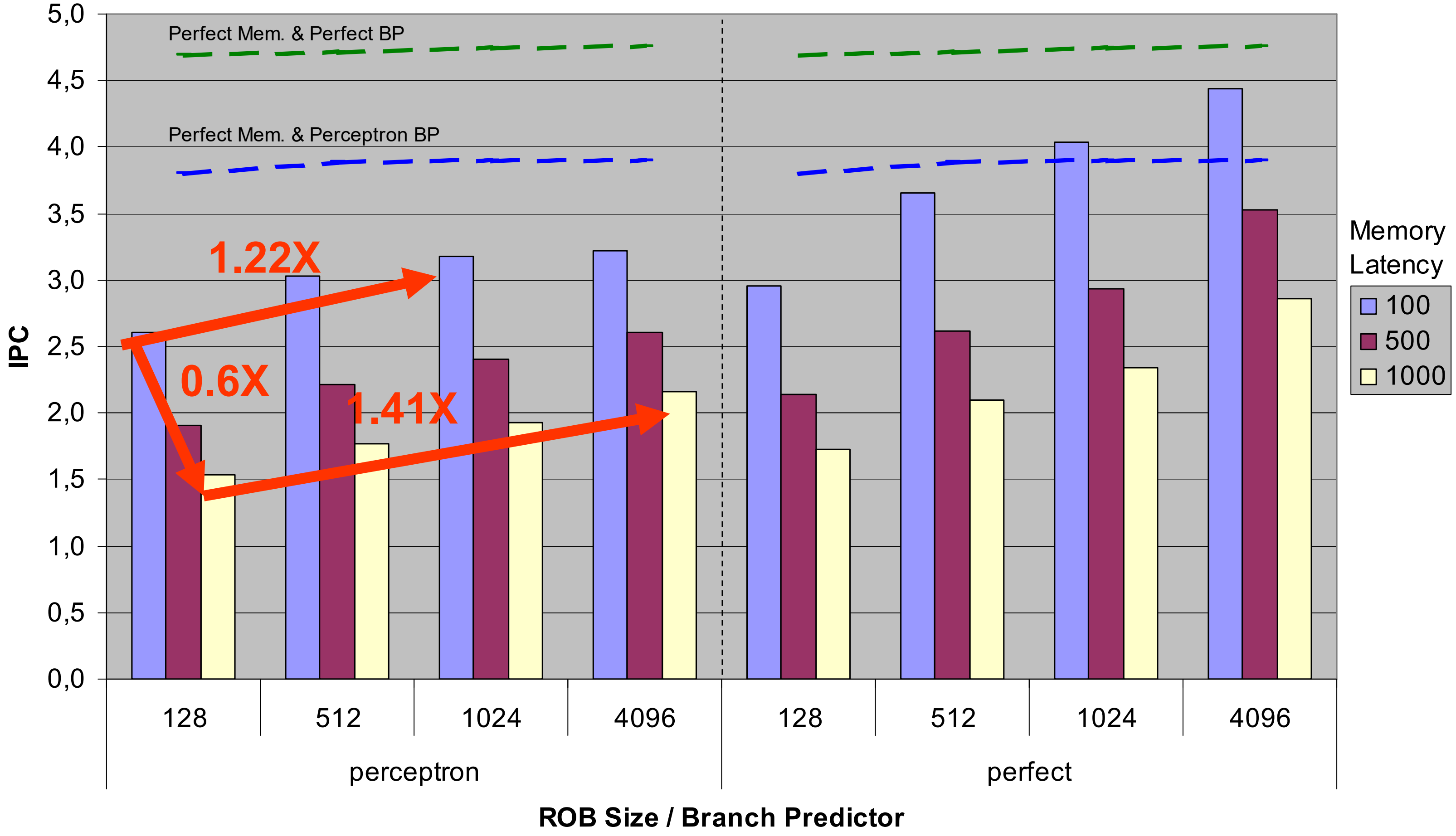
“Moore’s Law”

Processor-Memory Performance Gap: (grows 50% / year)

DRAM
7%/yr.

D.A. Patterson “ New directions in Computer Architecture” Berkeley, June 1998

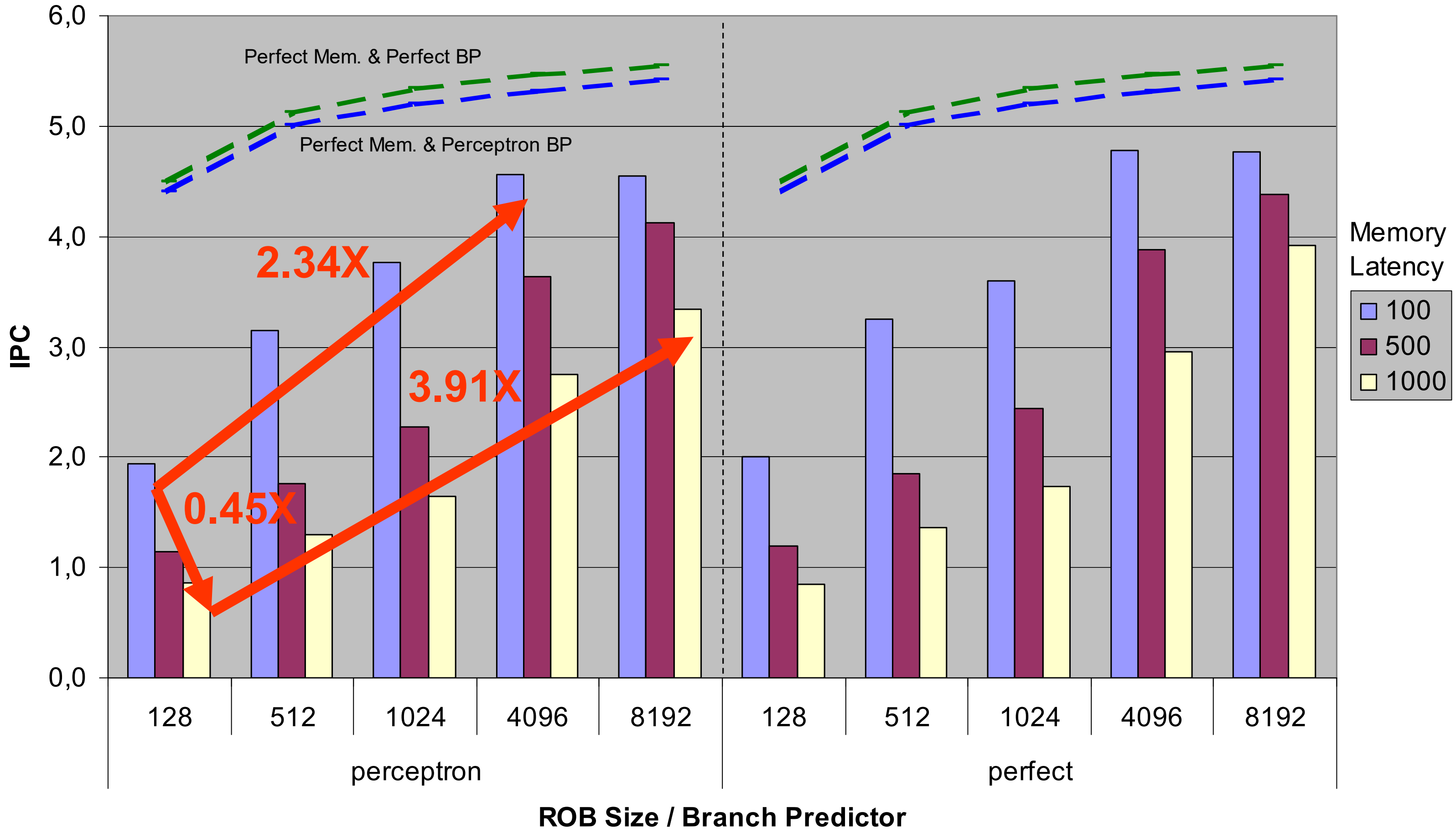
Integer, 8-way, L2 1MB



Research Proposal to Intel (July 2001) and presentation to Intel-MRL March 2002
Cristal et al.: "Large Virtual ROBs by Processor Checkpointing", TR UPC-DAC, July 2002
M. Valero. NSF Workshop on Computer Architecture. ISCA Conference. San Diego, June 2003

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Floating-point, 8-way, L2 1MB

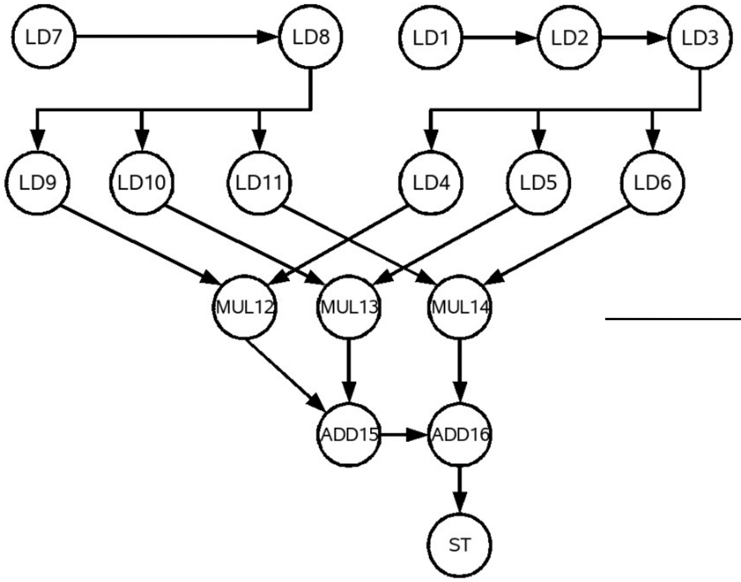


Research Proposal to Intel (July 2001) and presentation to Intel-MRL March 2002
Cristal et al.: "Large Virtual ROBs by Processor Checkpointing", TR UPC-DAC, July 2002
M. Valero. NSF Workshop on Computer Architecture. ISCA Conference. San Diego, June 2003

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

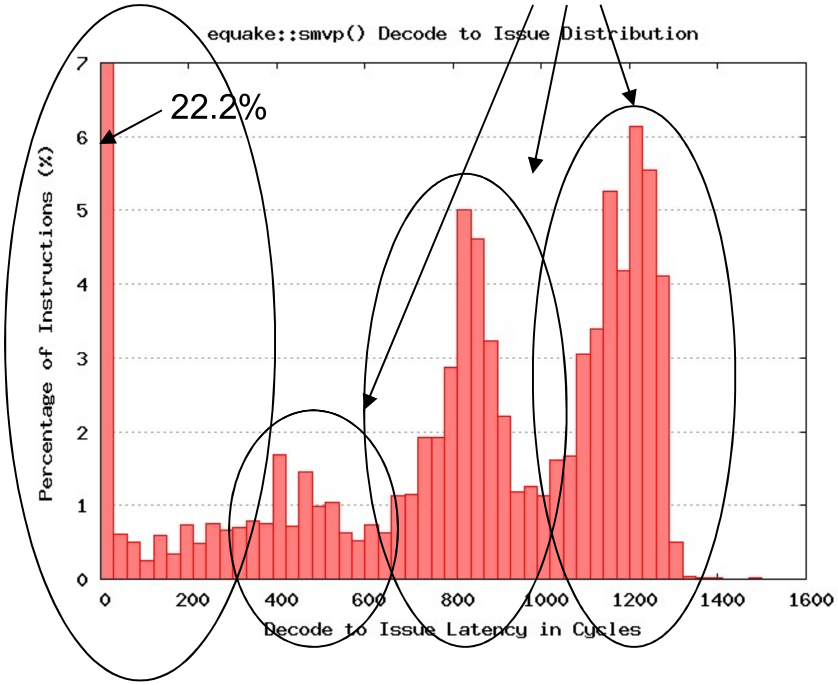
Execution Locality

```
void smvp(int nodes, double ***A, int *Acol,
                        int *Aindex, double **v, double **w) {
   [...]
    sum0 = A[Anext][0][0]*v[i][0] + A[Anext][0][1]*v[i][1] + A[Anext][0][2]*v[i][2];
   [...]
}
```



Cache-Dependent Code

Miss-Dependent Instruction Clusters



Mapping Clusters to Processors

- An execution cluster is a partition of the dynamic DDG belonging to the same locality group.

- **High Locality Clusters**:
  - large **amount** of instructions (70%) SpecFP, even more SpecINT
  - need to tolerate L2 cache hit latencies
  - advance as fast as possible (prefetching effect!)
  - thus the **Cache Processor** can be small, but must be **Out-of-Order**

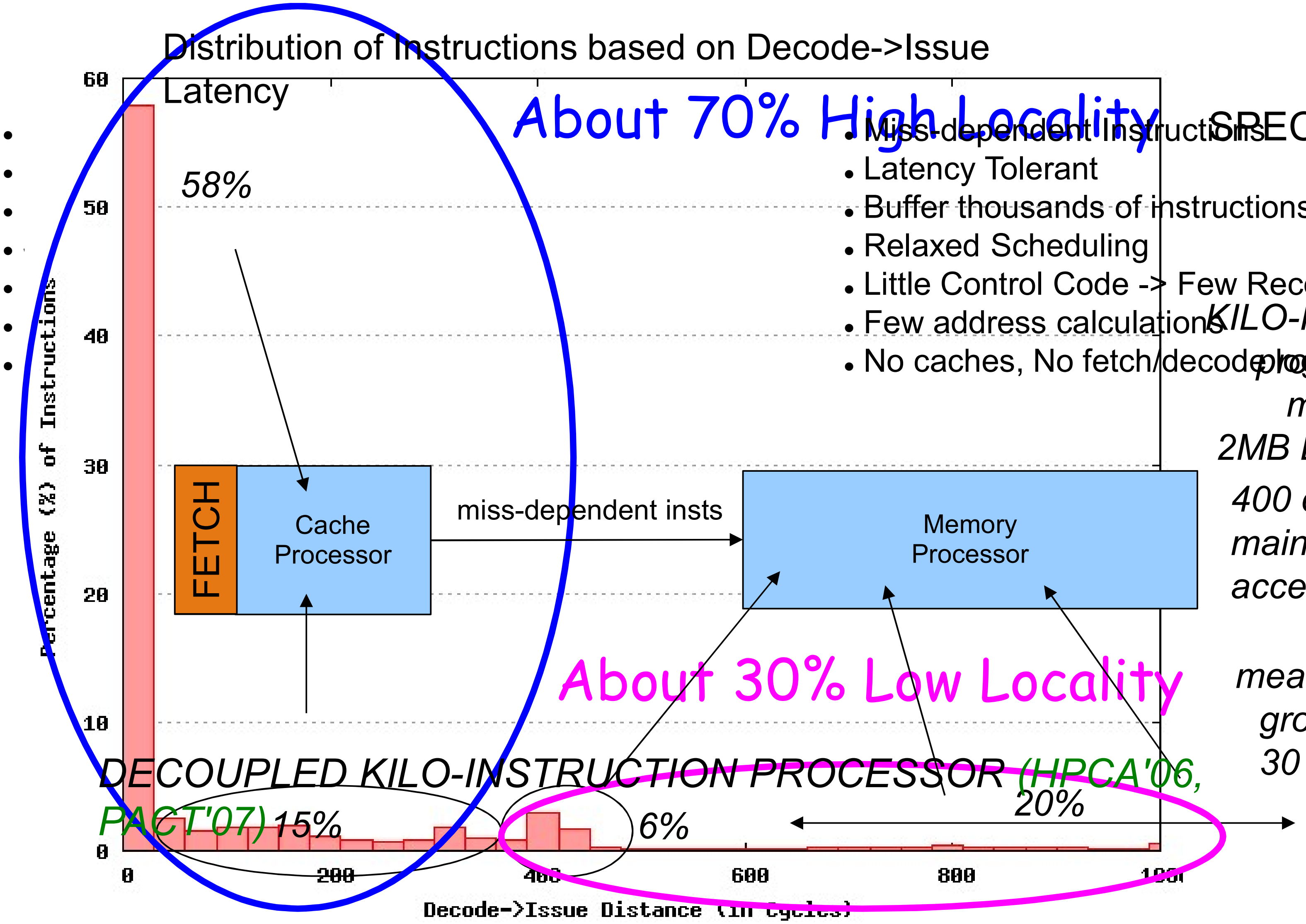
- **Low Locality Clusters**:
  - small amount of instructions (<30%)
  - generally not in critical path (Karkhanis, WMPI'02)
  - thus the **Memory Processor** can be even smaller, and probably **In-Order**

Slide 6

oo3 again LLC instead of L2?

osman.s.unsal osman.s.unsal, 7/2/2019

A different view: D-KIP



- Miss-dependent Instructions
- Latency Tolerant
- Buffer thousands of instructions
- Relaxed Scheduling
- Little Control Code -> Few Recoveries
- Few address calculations
- No caches, No fetch/decode logic

SPEC FP 2000

KILO-Instruction processor model

2MB L2 Cache

400 cycles main memory access latency

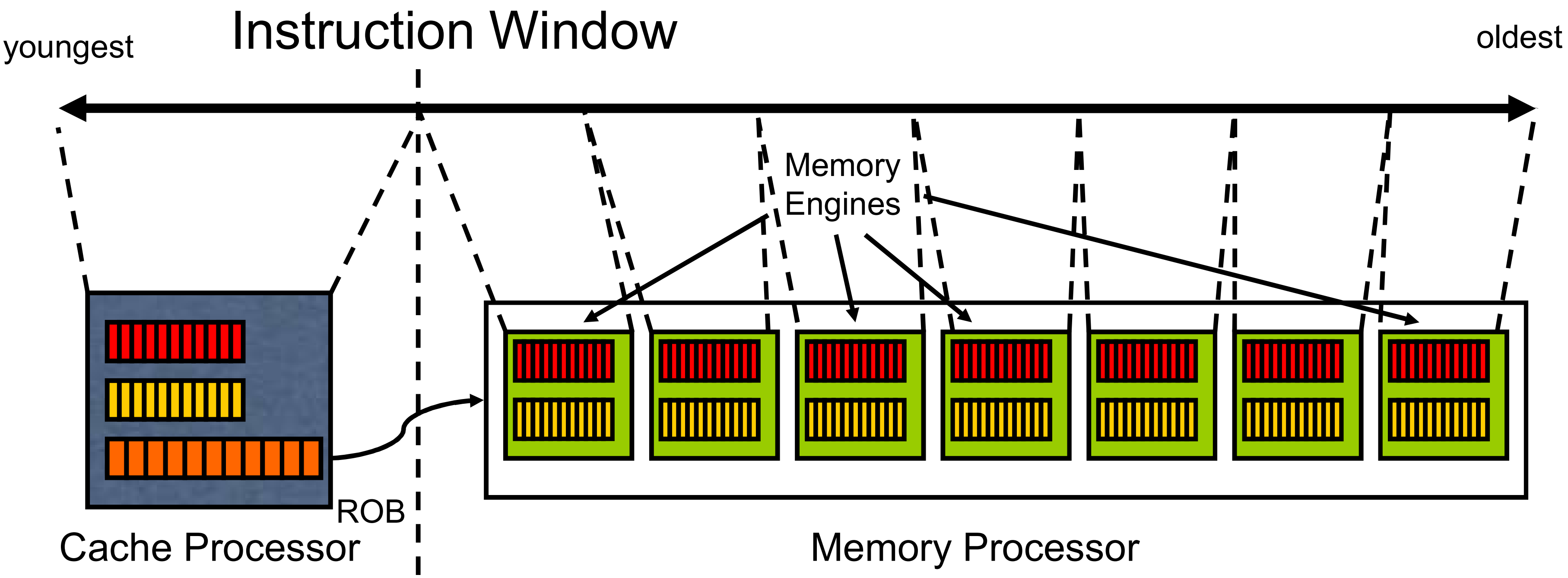
measured in groups of 30 cycles

DECOUPLED KILO-INSTRUCTION PROCESSOR (HPCA'06, PACT'07)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Miquel Pericas et al, “A decoupled KILO-instruction processor”, HPCA06

Flexible Heterogeneous MultiCore (I)



small out-of-order core designed assuming perfect L2. It processes all cache-dependent code

small dual-issue in-order processors executing code depending on L2 cache misses. Each memory engine processes a portion of the instruction window. Inclusion of all loads/stores provides sequential memory semantics. Activation and de-activation of *memory engines* changes the window size and allows to control power consumption

Cache-Dependent Code

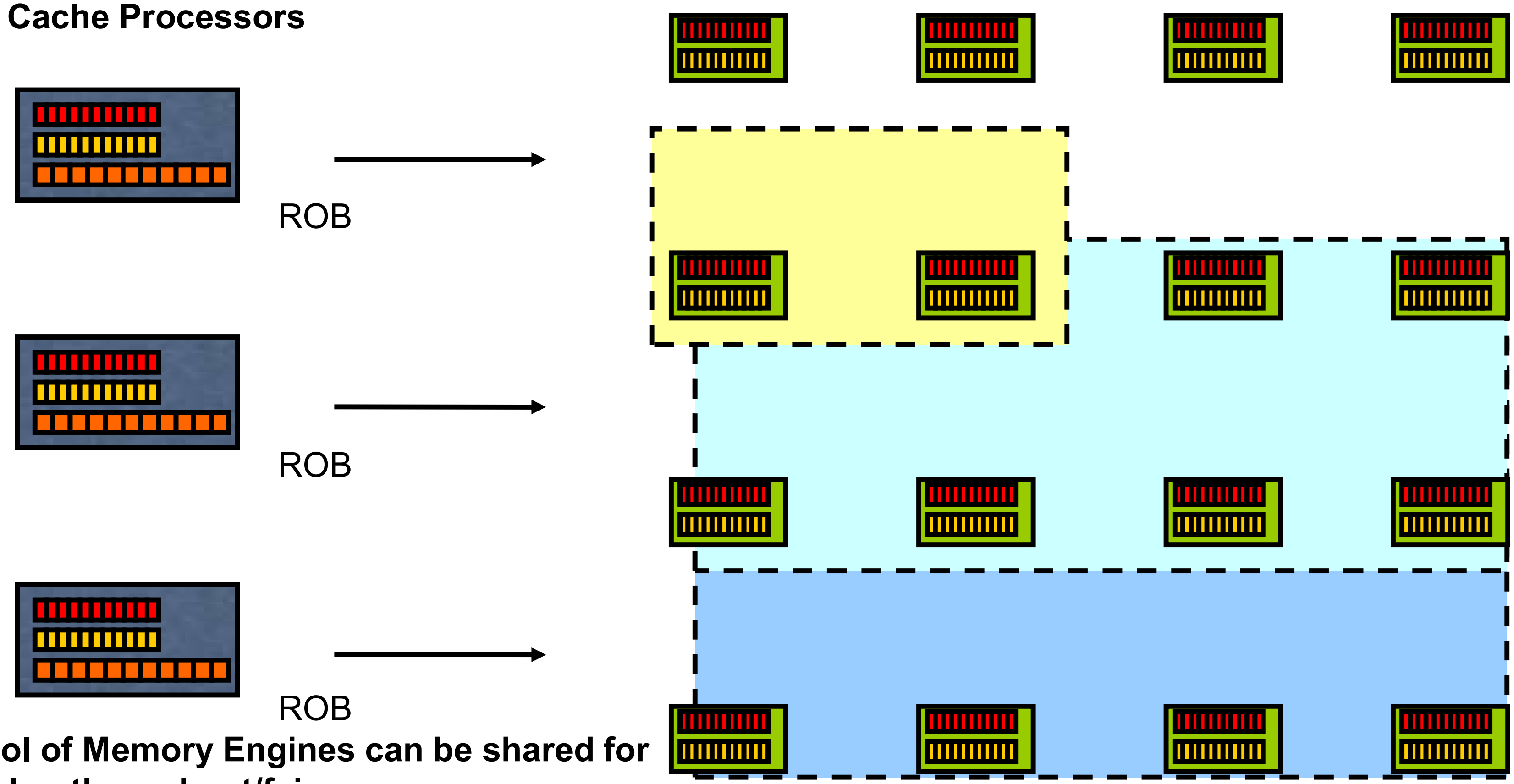
Miss-Dependent Code

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

Miquel Pericas et al, “A Flexible Heterogeneous Multi-Core Architecture”, PACT07

Flexible Heterogeneous MultiCore (II)

Extension to MultiThreading



Dynamically Assigned Pool of MEs

Cache Processors

ROB

ROB

ROB

Pool of Memory Engines can be shared for higher throughput/fairness

Kilo, Runahead and Prefetching

- Prefetching
  - Anticipates memory requests
  - Reduces the impact of misses in the memory hierarchy

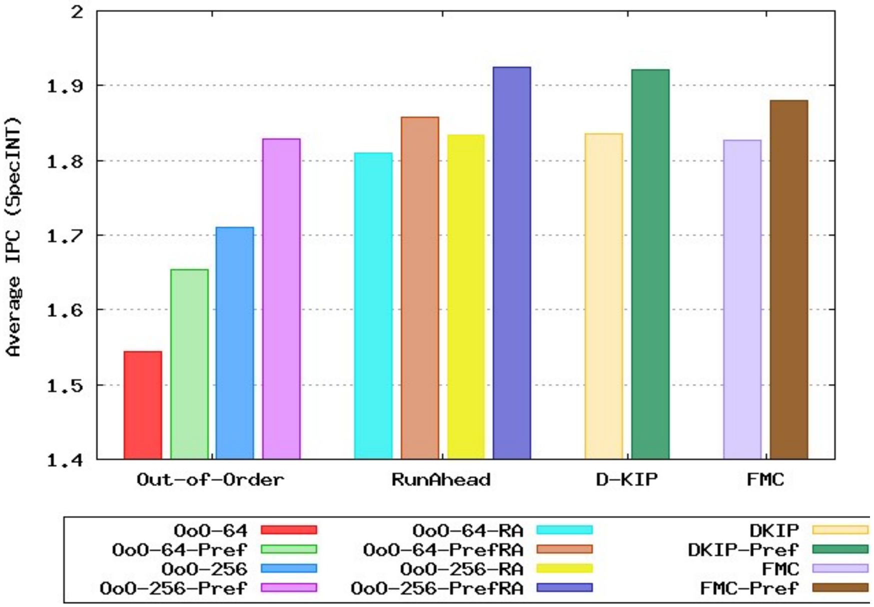
- Runahead Mechanism
  - Executes speculative instructions under a LLC miss
  - Prevents the processor from stall when the ROB is full
  - Allows generating useful data prefetch

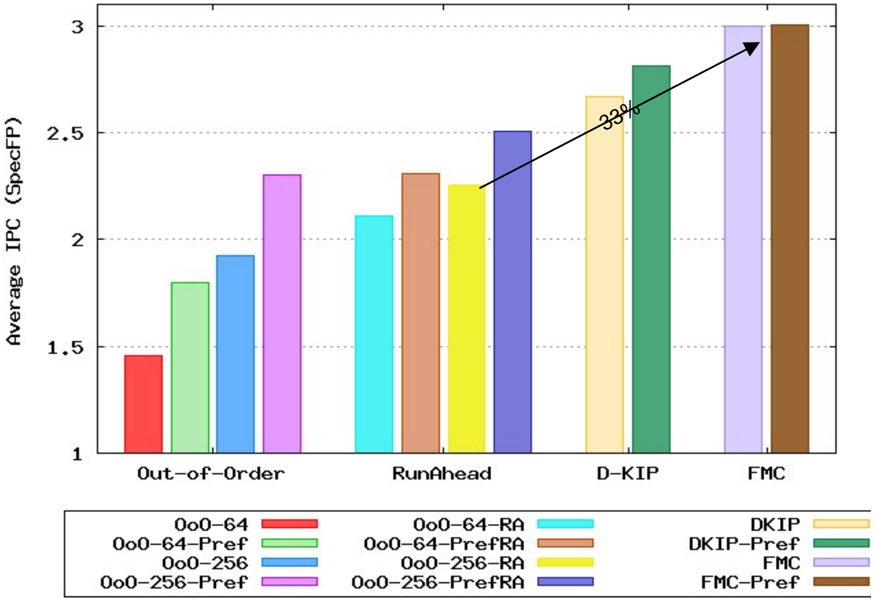
- Kilo-instruction Processors
  - Exploits more ILP by maintaining thousands of in-flight instructions while long-latency loads are outstanding in memory (implicit prefetching)

Tanausú Ramírez et al., "Kilo-instruction Processors, Runahead and Prefetching", CF'06, May, 2006.

Performance versus RunAhead and Stride Prefetching

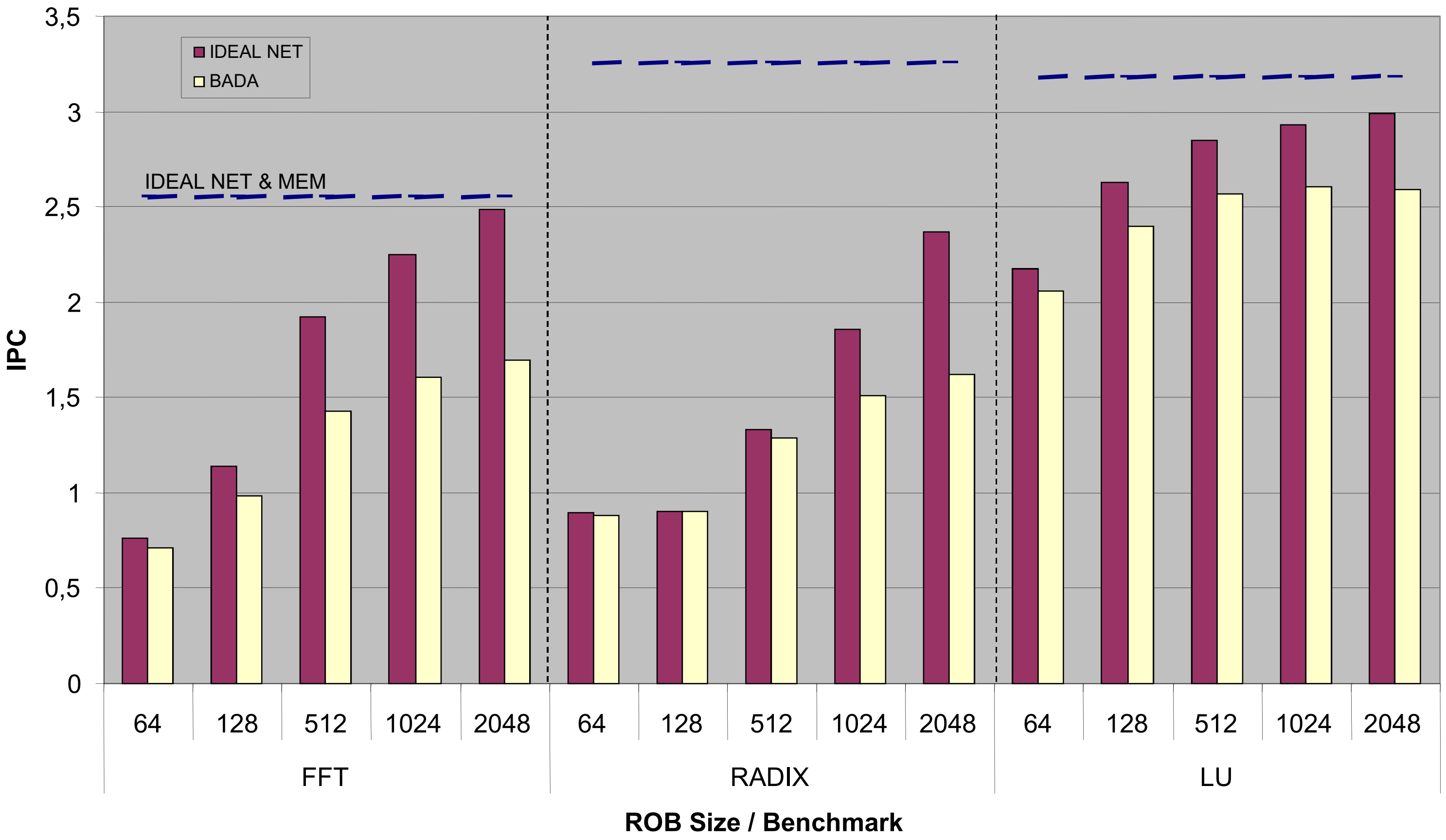
- OoO and RunAhead are 4-way with 64/256-entry ROBs
- Cache Processors are 4-way with 64-entry ROB
- Memory Processor/Memory Engines are two-way in-order processors
- A Memory Engine can hold up to 128 long-latency instructions and 128 loads/stores
- RunAhead features ideal runahead cache
- Stream prefetcher can hold up to 64KB of prefetched data





BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

“Kilo-processor” and multiprocessor systems



M. Galluzzi et al. “ A First glance at Kiloinstruction Based Multiprocessors” Invited Paper. ACM Computing Frontiers Conference. Ischia, Italy, April 10-12, 2004

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

What we wanted to do

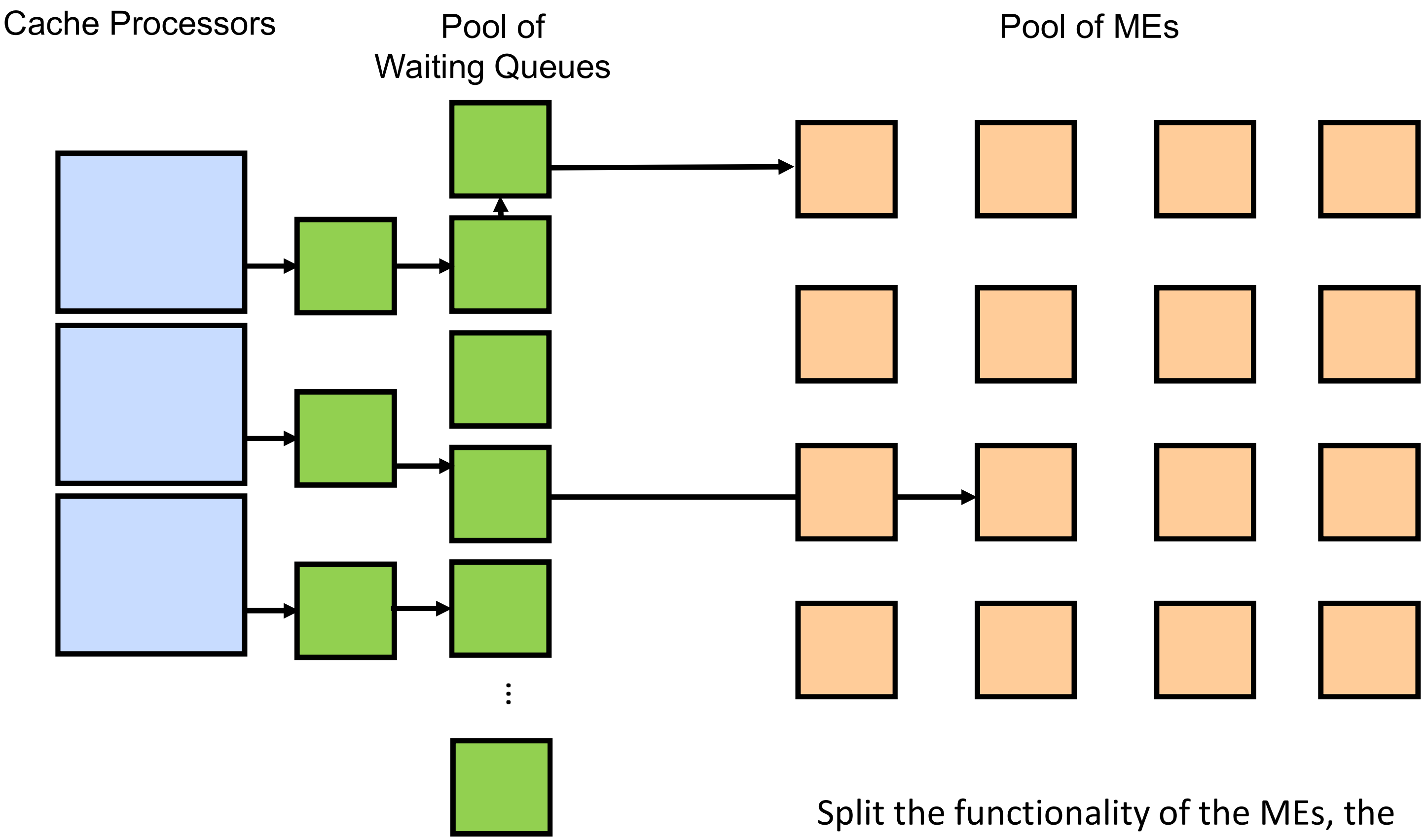
- Can we extend a Big-Little multicore to implement the FMC?

- Are the Memory Engines (Mes) used all the time or are they waiting for long latency loads?

- Can we do something to avoid discarding all the MEs in case of branch mispredictions?

- How does a practical kilo-vector processor look like?

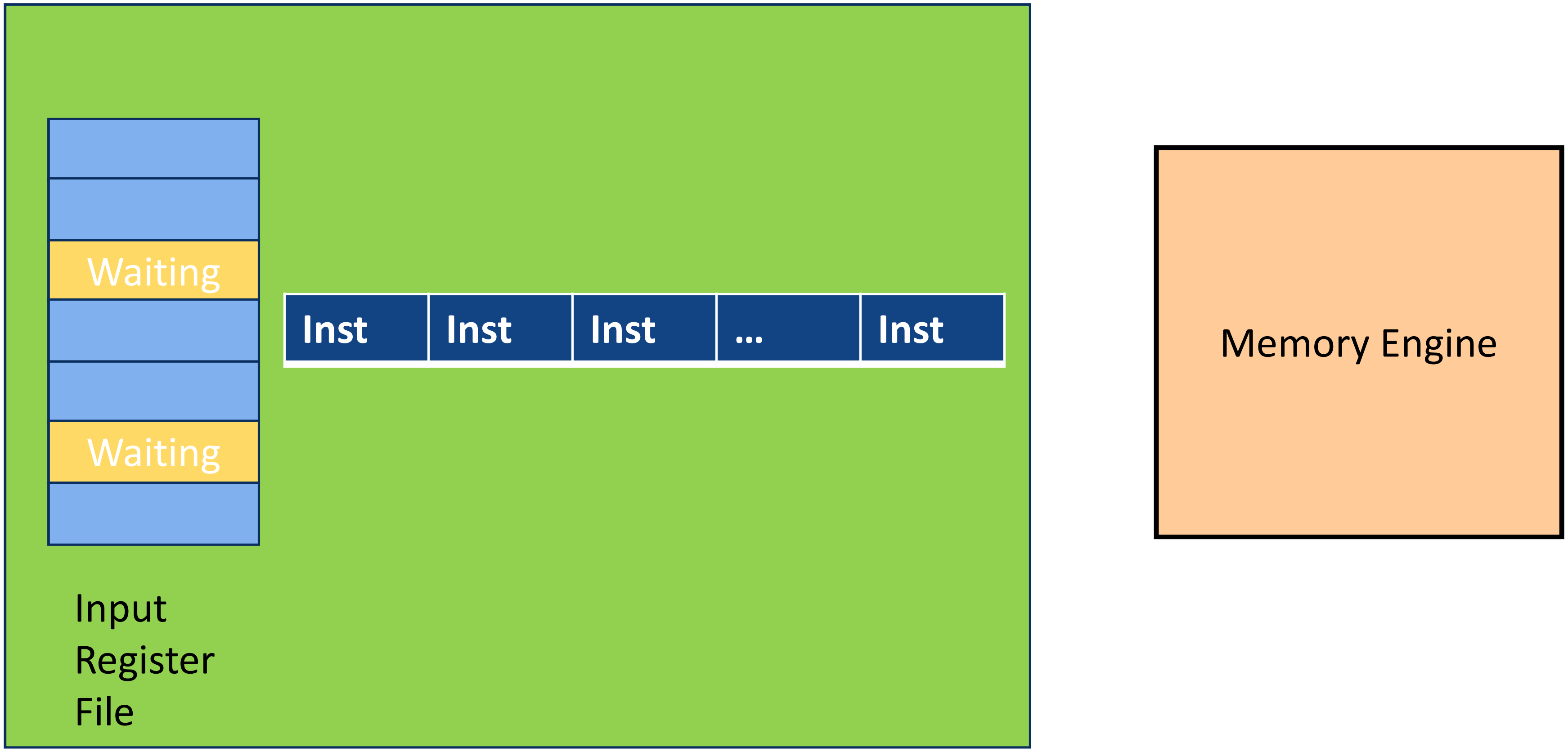
Some ideas “stolen” from “Edge Processors” and “Decoupled Architectures”



Split the functionality of the MEs, the instruction queues and the functional units

Waiting Queue

- Instructions + Logical Registers
  - Wait until all used logical registers are ready
  - Assign a Memory Engine



Waiting

Waiting

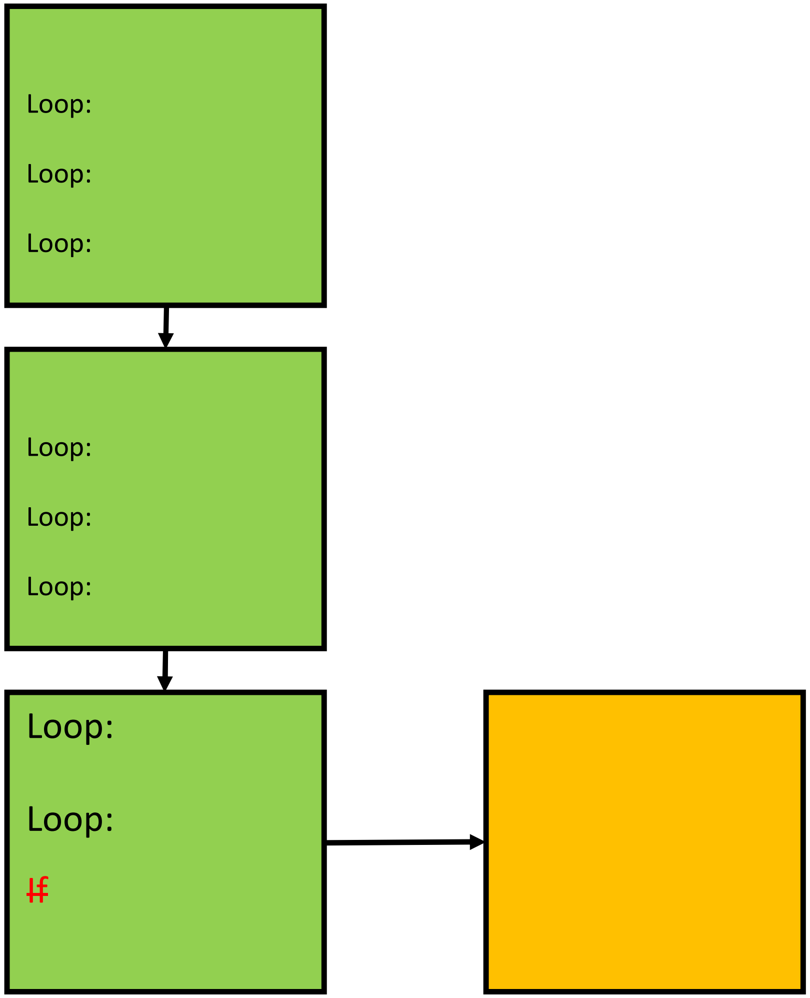
Inst | Inst | Inst | … | Inst

Input Register File

Memory Engine

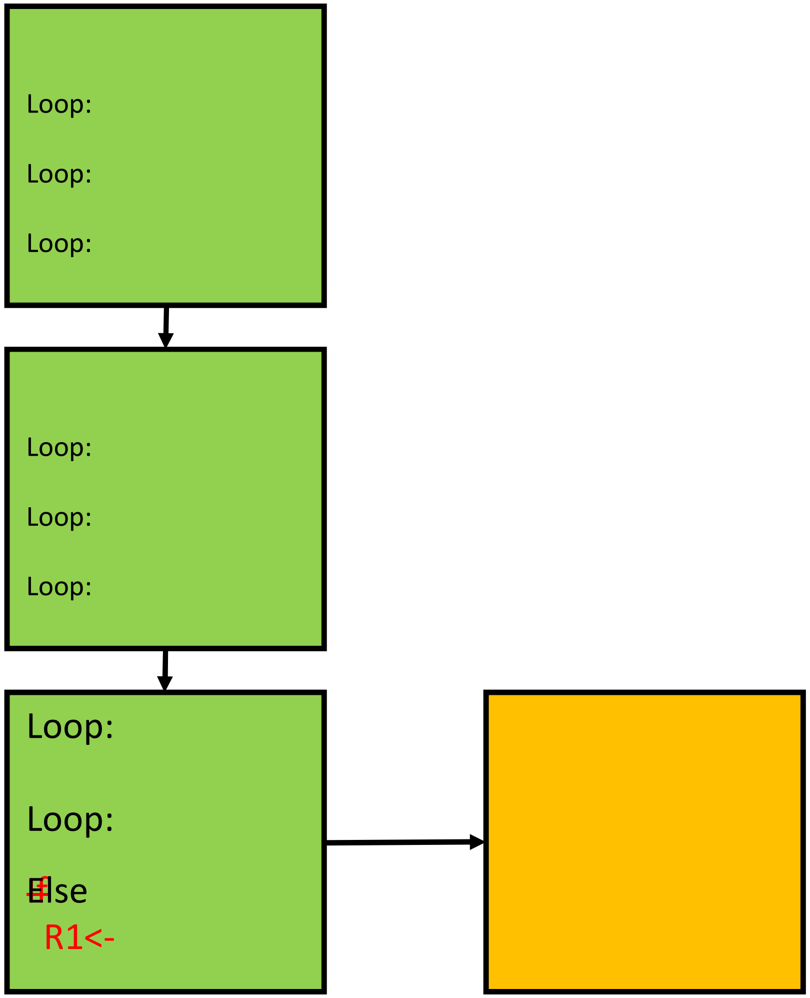
Where to start a waiting queue

- Loop:
- …
- …
- If: Br …
- …
- …
- Else: …
- …
- …
- Fi:
- …
- Endloop



Where to start a waiting queue

- Loop:
- …
- …
- If: Br …
- …
- …
- Else: …
- …
- …
- Fi:
- …
- Endloop



Loop:

Loop:

Loop:

Loop:

Loop:

Loop:

Loop:

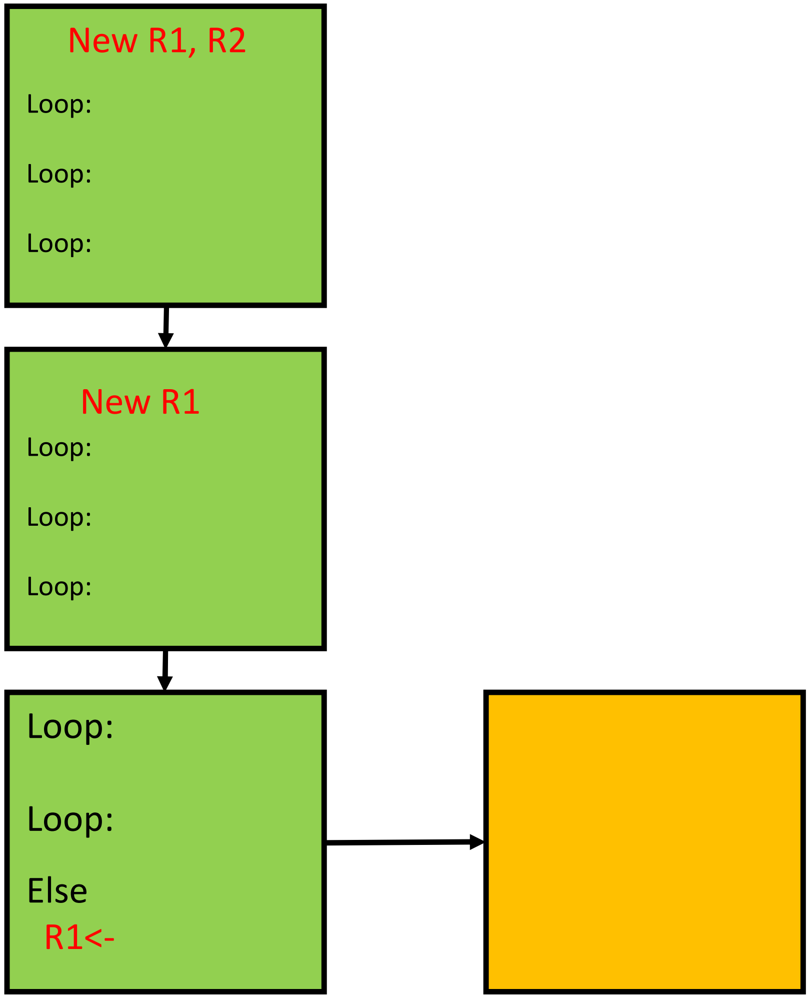
Loop:

Else

R1<-

Where to start a waiting queue

- Loop:
- …
- …
- If: Br …
- …
- …
- Else: …
- …
- …
- Fi:
- …
- Endloop



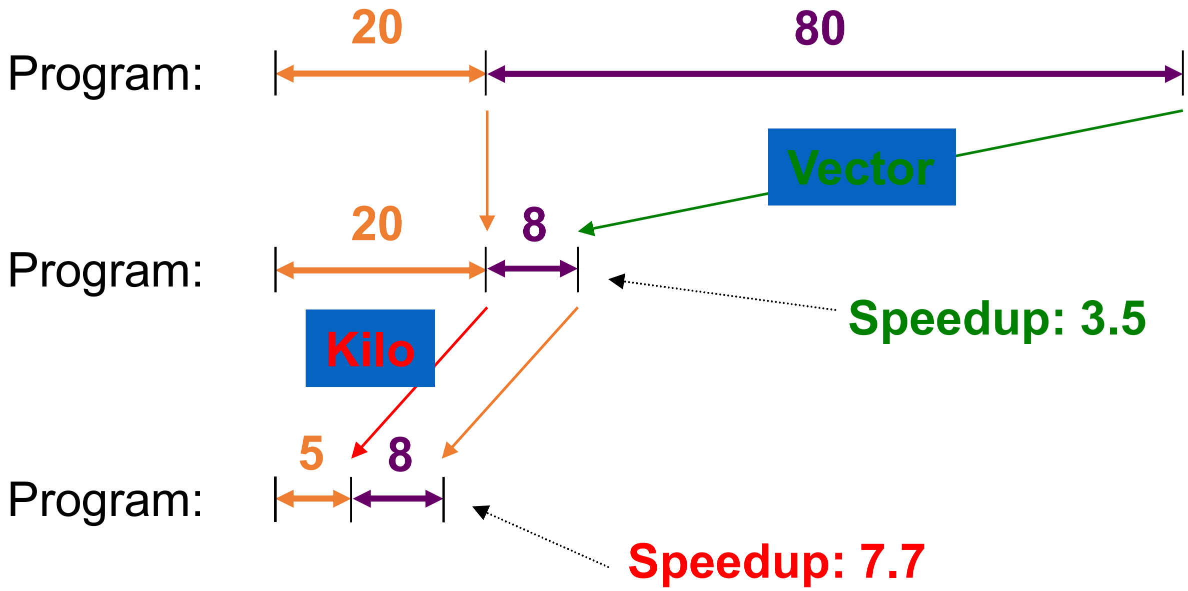
Problems and more problems

- What to do if the addresses of Loads and Stores are modified

- Fetching instructions, and partial reexecution

- Pointer Chasing
  - Start a new waiting queue or suspend the execution of a waiting queue?

“Kilo-vector” processor



F. Quintana et al, “Kilo-vector” processors, UPC-DAC

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

EXCELENCIA
SEVERO
OCHOA

Thank you

Adrian.cristal@bsc.es