Exploring the use of light threads to improve the instruction level parallelism

D. González Márquez[1] A. Cristal Kestelman[2] E. Mocskos[1]

[1]Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires (C1428EGA), Argentina.

[2]Barcelona Supercomputing Center, Artificial Intelligence Research Institute - CSIC, Barcelona (08034), Spain.

Mendoza, 30/7/2013

Introduction

- The research in **processor architectures** centers in optimizing the processor design based on *performance*, *consumption*, *production cost*, *surface* etc.

Introduction

- The research in **processor architectures** centers in optimizing the processor design based on *performance*, *consumption*, *production cost*, *surface* etc.

- Parallelism allows the hardware to **accelerate applications** by executing *multiple, independent operations concurrently*.

Introduction

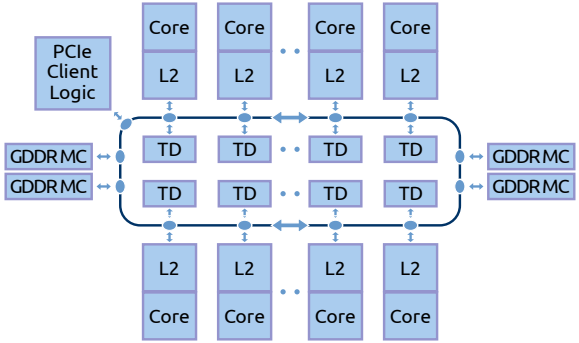
- The research in **processor architectures** centers in optimizing the processor design based on *performance*, *consumption*, *production cost*, *surface* etc.

- Parallelism allows the hardware to **accelerate applications** by executing *multiple, independent operations concurrently*.

- Three levels:
  - *instruction-level parallelism* (**ILP**)
  - *thread-level parallelism* (**TLP**)
  - *data-level parallelism* (**DLP**)

More cores are coming

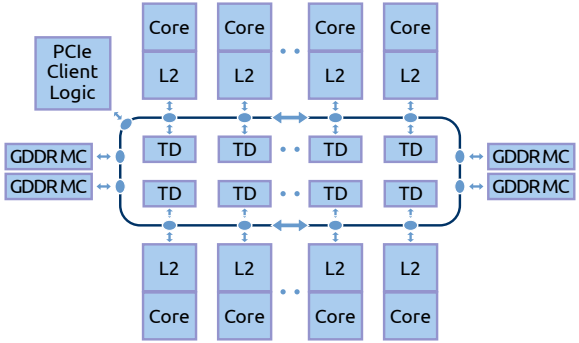
The improvements in processor technology give us more avaliable cores



http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner

More cores are coming

The improvements in processor technology give us more avaliable cores

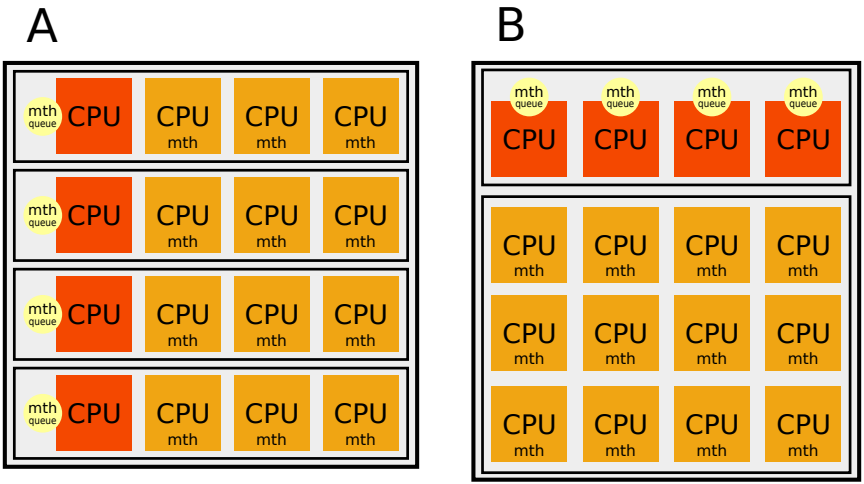


http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner

But...

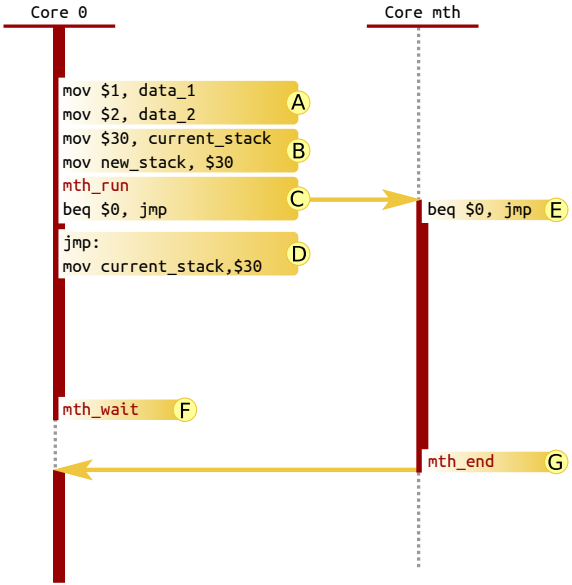
The applications and the programming models are not prepare for hundreds of cores.

We propose a change in the whole system



Two types of cores: full controlling cores and simple cores.

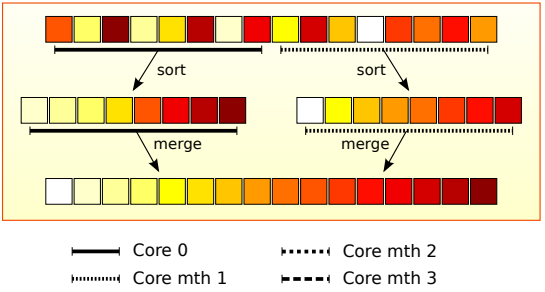
Simple example of interaction



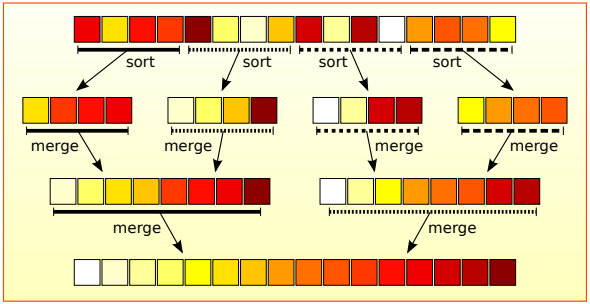
Testing the idea - Sorting an array using merge-sort

The new processor is completely simulated with `gem5` based on `ALPHA` architecture.
Instructions added: `mth_run`, `mth_delegate`, `mth_end` and `mth_syn`.

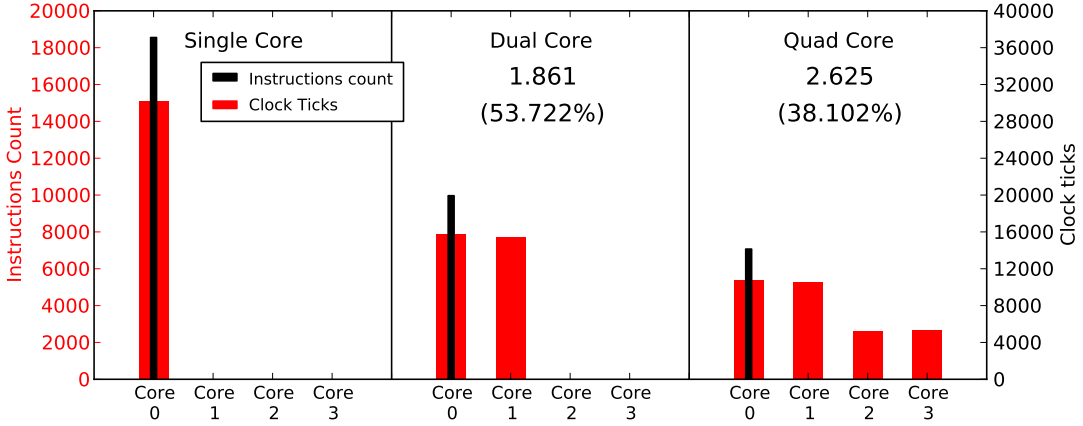
2 CORE EXAMPLE



4 CORE EXAMPLE

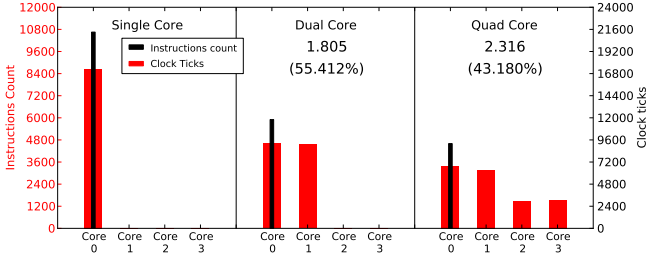


Results with 100 elements

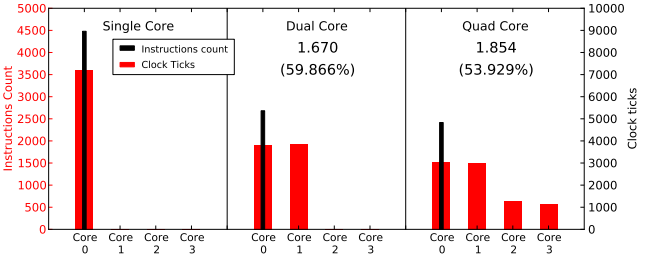


The unbalanced load in the four cores case comes from the implemented algorithm: two cores remain idle while the other two compute the last stage of the merge.

But we can deal with smaller problems



64 elements



32 elements

Conclusions and future work

- The proposal can effectively deal with very small problems.
- Standard tools need larger problems.
- Can execute small pieces of code in different simple processors.

Conclusions and future work

- The proposal can effectively deal with very small problems.
- Standard tools need larger problems.
- Can execute small pieces of code in different simple processors.
- New **processor organization** and **programming model**.
- Need to further test with **more processors**.
- Need to add support from **compilers** and **operating system**.

Thanks!

Questions?