Parallel Computing Paradigms

João Luís Ferreira Sobral
www.di.uminho.pt/~jls
jls@...

Web: Elearning

Parallel Computing Paradigms

At the end of the course, students should be able to:

- **Design** and **optimise** parallel applications that can **efficiently** run on a wide range of parallel computing platforms
  - Identify/develop parallel applications using well-known parallelism patterns
  - Identify limitations of current parallel programming paradigms and languages
  - Identify limitations and workarounds to performance scalability

Program (short version)

- **Programming models and languages**
  - Shared vs distributed memory models
  - Threads, communicating processes and distributed objects
  - Tasks distribution among resources
  - Mechanisms to express concurrency/parallelism
- **Design of parallel applications**
  - Design phases: partition, communication, agglomeration, mapping
  - Typical parallel algorithms: *pipelining*, *farming*, *heartbeat* e *divide & conquer*
  - Measurement, analysis and optimisation of parallel applications
  - Scalability analysis: cost / benefit of parallelism and its quantification (metrics)
  - Granularity control: computation versus communication
  - Load distribution techniques and data partition

Parallel Computing Paradigms

Final grade

- One mini-project: development of typical parallel applications
  - On shared memory (35%)
  - On distributed memory (50%)
  - On hybrid SM+DM(15%)

Requirements

- Basic Java and C /SCD

Bibliography (base)

- Slides
- M. Quinn. Parallel programming in C with C and OpenMP, McGraw Hill, 2003
- I. Foster. *Designing and Building Parallel Programs*, Addison-Wesley, 1995.

Bibliography (additional)

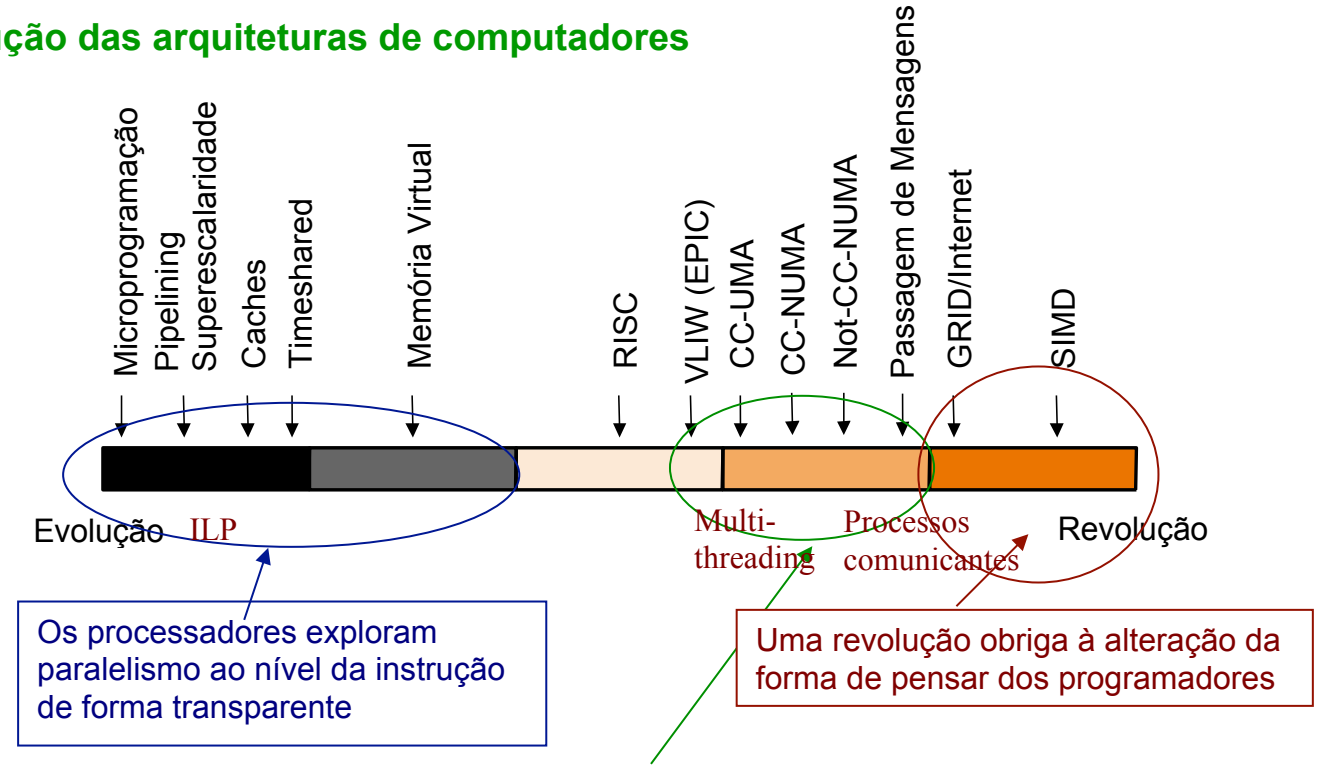
- M. McCool, J. Reinders, A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*, Morgan Kaufmann, 2012
- C. Lin, L. Snyder, *Principles of parallel programming*, Pearson international,2009
- R. Gerber, A. Binstock. *Programming with Hyper-Threading Technology,* Intel Press, 2004.
- F. Buschmann, D. Schmidt, M. Stal, H. Rohnert, *Pattern-oriented Software Architecture Vol 2: Patterns for Concurrent and Networked Objects*, John Wiley and Sons Ltd, 2000.

Parallel Computing Paradigms

- Introdução
  - Evolução das arquiteturas de computadores
  - Níveis e tipos de paralelismo
  - Paradigmas vs linguagens de programação
  - Linguagens baseadas em: diretivas (OpenMP), extensões à linguagem (Java) vs bibliotecas (TBB)
- Programação baseada no paradigma de memória partilhada
  - OpenMP
  - Java threads / cilk
  - Intel *Thread Building Blocks*
  - Modelos de consistência de memória
  - Medição de otimização de desempenho em memória partilhada
- Programação baseada no paradigma de memória distribuída
  - Processos comunicantes (MPI)
  - Objetos distribuídos, cliente/servidor
  - Desenho de aplicações paralelas
  - Padrões comuns de computação
  - Medição e otimização de desempenho em memória partilhada
- Programação baseada em modelos híbridos
  - Modelo híbrido MPI + OpenMP
  - Modelos de memória virtual partilhada (*virtual shared memory*)
  - Modelo global com partições (*partitioned global address space*)
- Modelos e paradigmas de computação paralela
  - classificação das várias linguagens para computação paralela

Paradigmas de Computação Paralela

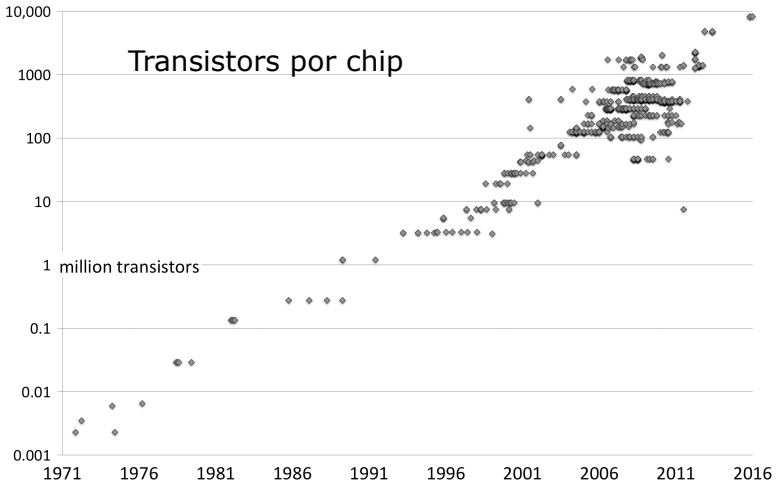
Evolução das arquiteturas de computadores

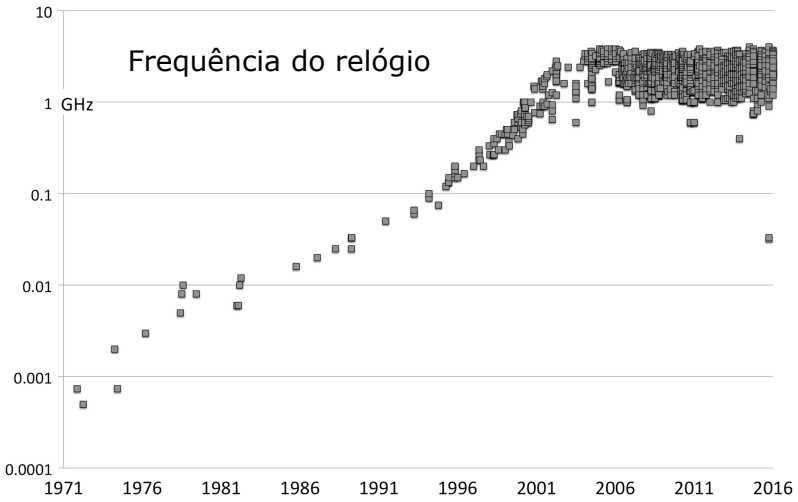


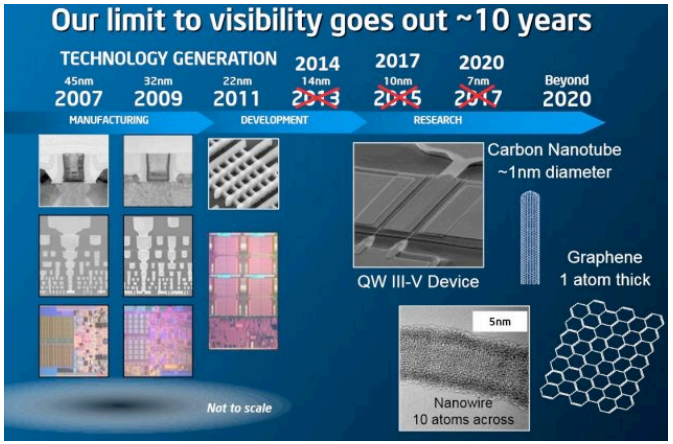
A ênfase de computação paralela é na programação deste tipo de arquitecturas

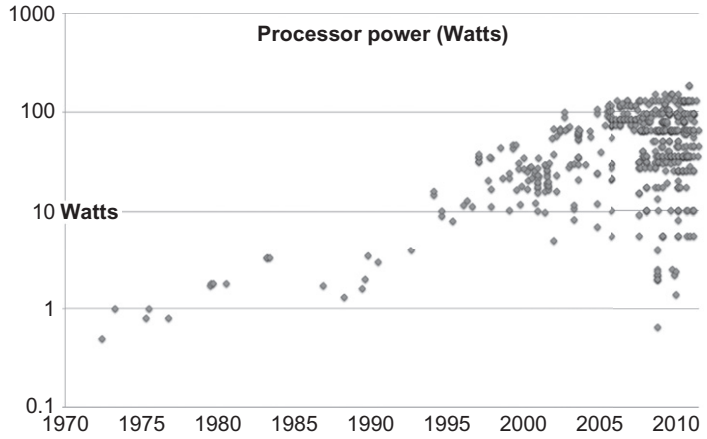
Paradigmas de Computação Paralela

Evolução das arquiteturas de computadores (nota: escala lin-log)



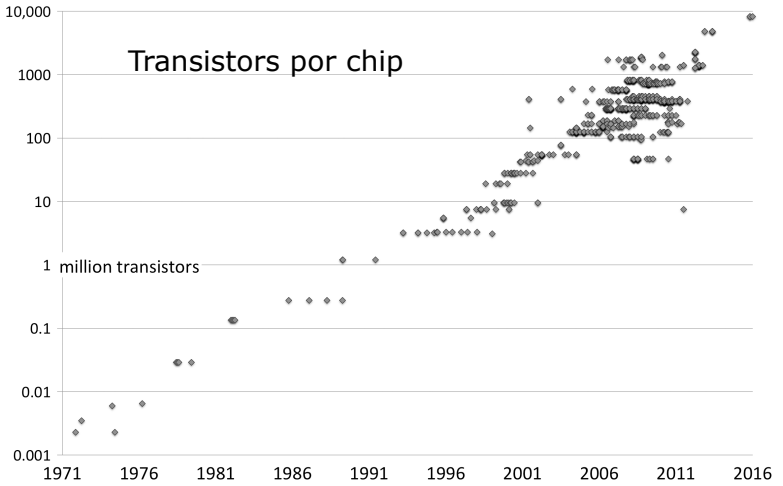


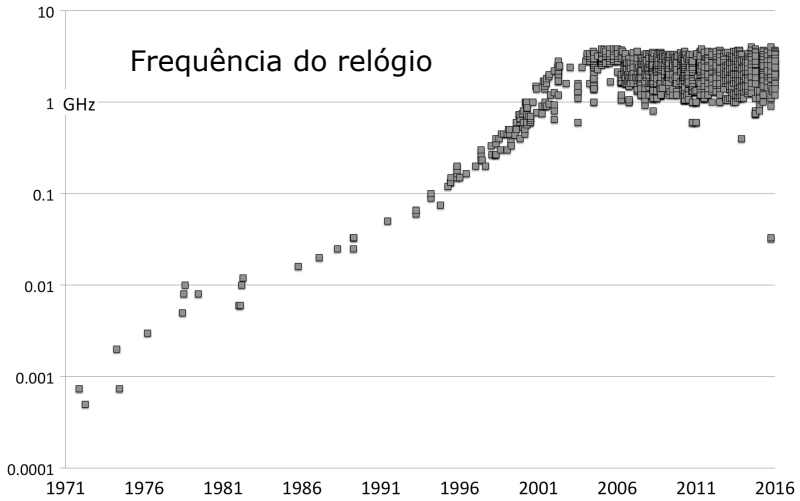


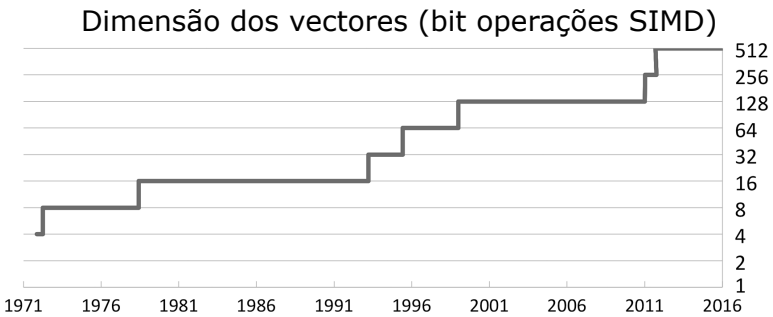


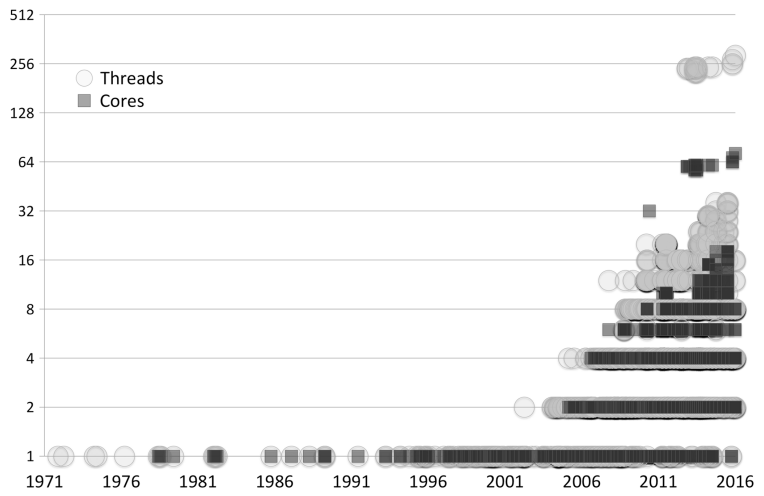
Paradigmas de Computação Paralela

Evolução das arquiteturas de computadores (nota: escala lin-log)









Paradigmas de Computação Paralela

Arquiteturas paralelas

	Base PS4
CPU	Eight Jaguar cores clocked at 1.6GHz
GPU	18 Radeon GCN compute units at 800MHz
Memory	8GB GDDR5 at 176GB/s

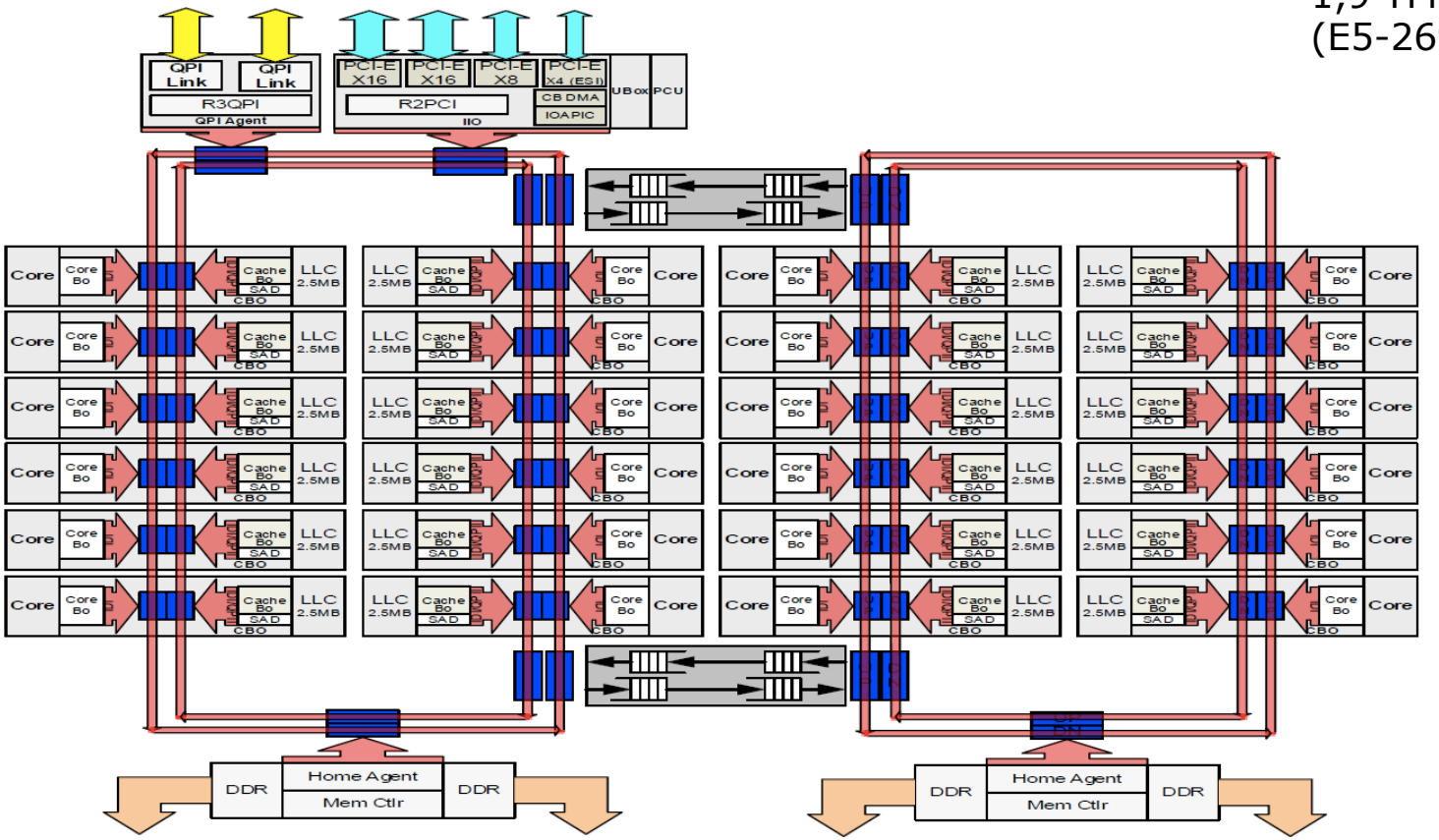
0,10 TFlop/s

1,84 TFlop/s

Quad Core **Apple A10 Fusion** Processor with embedded M10 Motion coprocessor, Hexa-Core Graphics chip
2GB of RAM

Intel Xeon-E5 v4 (max 24 cores – 48 *threads*)

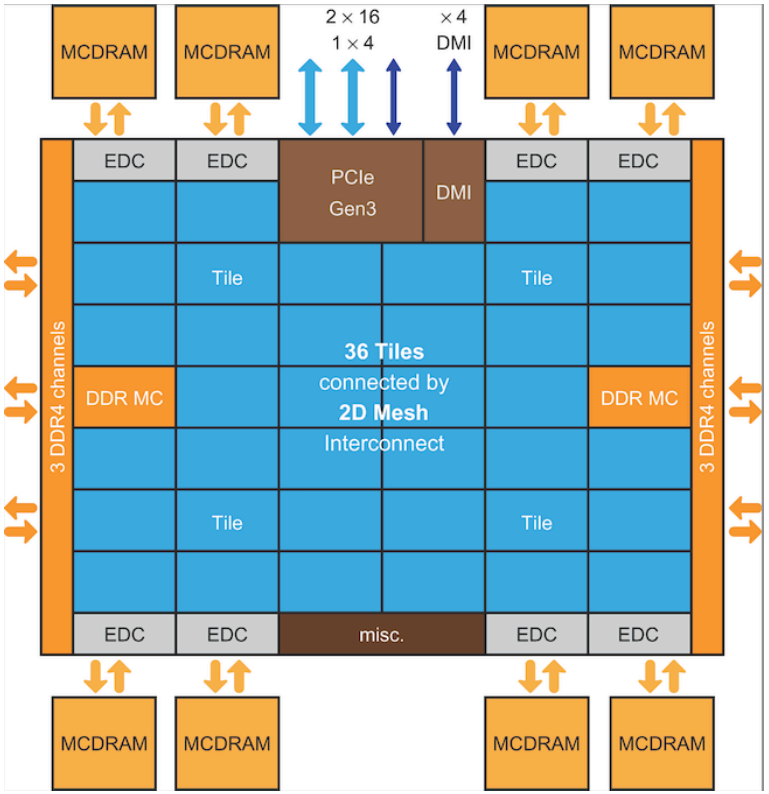
1,9 TFlops
(E5-2699v4)



Arquiteturas paralelas – Knights Landing

- até 72 cores/chip
- 4 threads/core
- 1MB L2 partilhada por “tile” (dois cores)
- Duas unidades vetoriais de 512b/core
- 16GB MCDRAM (500GB/sec)
- 200W TDP

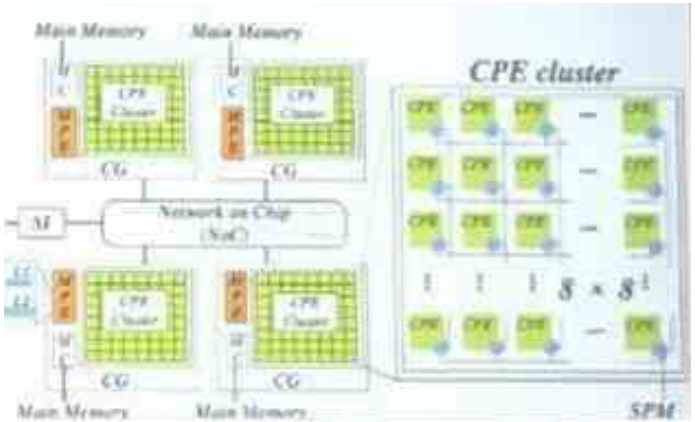
- 3 TFlop/s

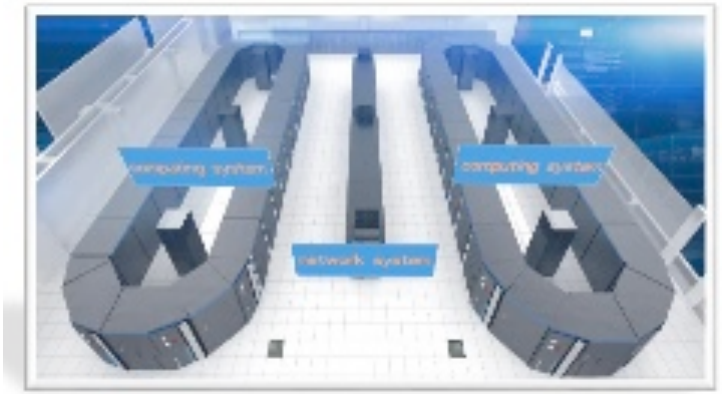


Arquiteturas paralelas – Sunway TaihuLight

- Nº1 do top 500 (www.top500.org)
- 10 649 600 cores (124,4 Pflop/s)
- 15,3 MW (refrigeração a água)

- Processador SW26010
  - 4x 64 cores por chip
  - 1,45 GHz
  - 64KB Scratch Pad Memory (SPM)

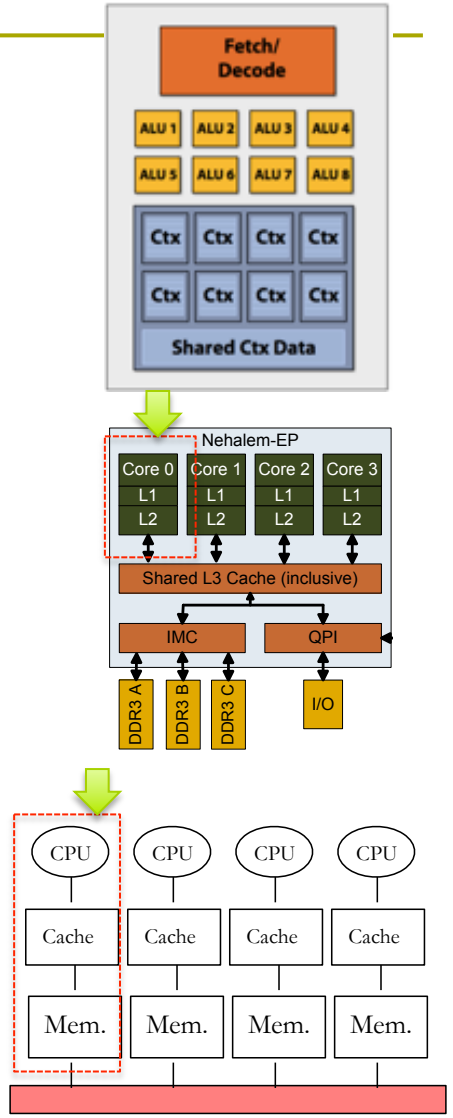




Paradigmas de Computação Paralela

Níveis de paralelismo (HW+SW)

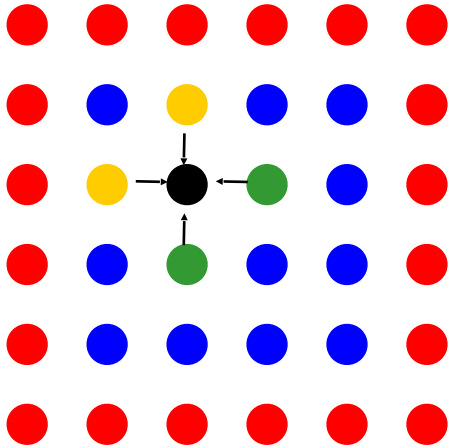
- **Instrução (ILP)**
  - Execução de múltiplas instruções de um programa em paralelo
  - Processamento vetorial
  - Explorado pelo hardware atual
  - Limitado pelas dependências de dados/controlo do programa
- **Tarefas / fios de execução**
  - múltiplos fluxos de instruções de um mesmo programa executam em paralelo
  - Limitado pelas dependências e características do algoritmo
- **Processos**
  - Múltiplos processos de um mesmo programa / ou de vários programas



Paradigmas de Computação Paralela

Níveis de paralelismo: exemplo "stencil"

$$A[i,j] = 0.2 \times (A[i,j] + A[i,j-1] + A[i-1,j] + A[i,j+1] + A[i+1,j])$$

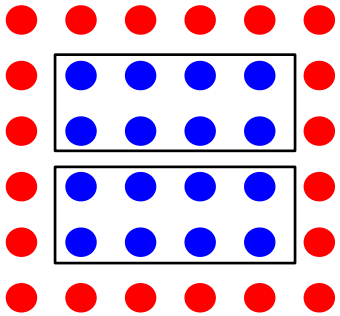


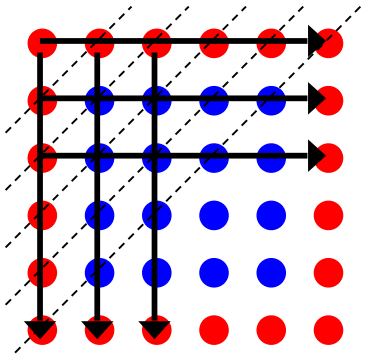
- **Instrução (ILP)**
  - Ler valores A[..,..] da memória em paralelo
  - Efetuar as operações aritméticas em paralelo
  - Multiplicação por 0,2 e escrita de A[i,j] só no final do cálculo
  - **Calcular valores de A em paralelo?**

Níveis de paralelismo: exemplo "stencil"

$$A[i,j] = 0.2 \times (A[i,j] + A[i,j-1] + A[i-1,j] + A[i,j+1] + A[i+1,j])$$

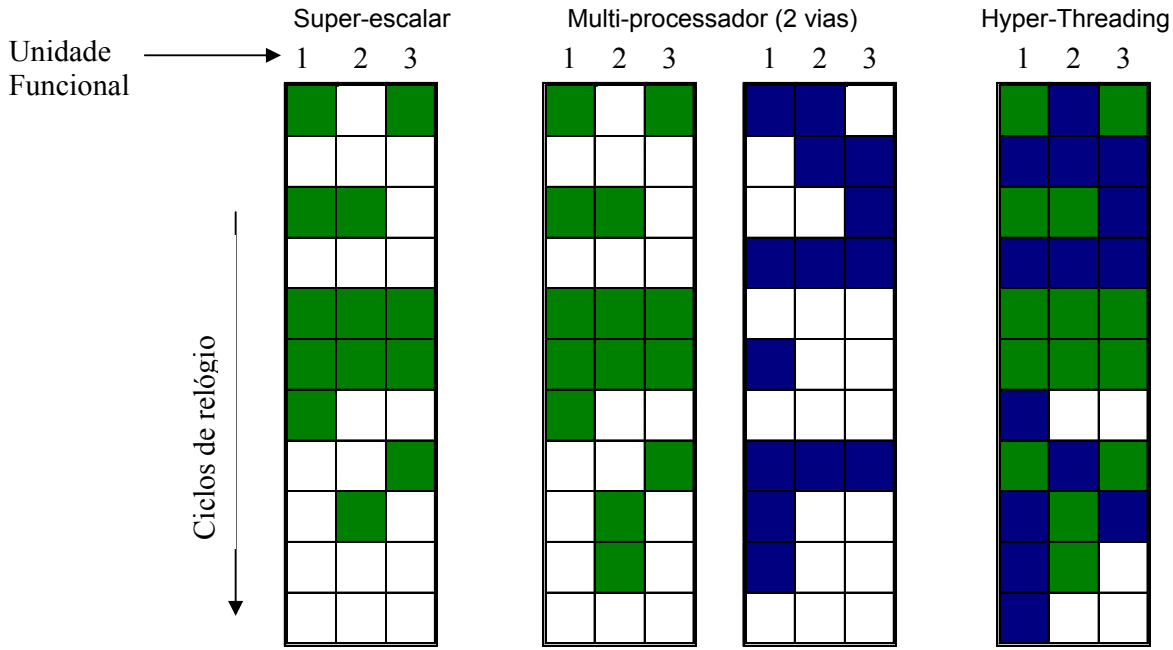
- **Fios de execução**
  - Cada atividade calcula uma parte dos valores da matriz
  - Dependências?





Níveis de paralelismo: software VS hardware

- **Hyper-threading: Fios de execução usados para aumentar o ILP**

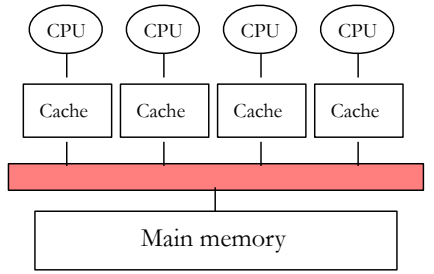


1 processador = 2 processadores ?
Não porque parte dos recursos do processador não são duplicados (i.é., são divididos pelos vários fios de execução)
(*caches*, registos internos, *buffers* internos, etc.)

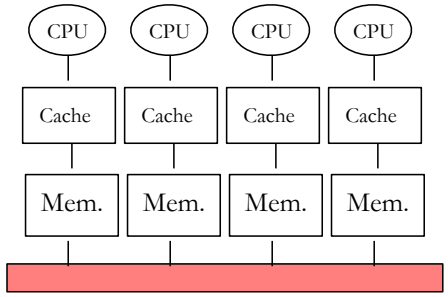
Paradigmas de Computação Paralela

Memória partilhada VS memória distribuída (HW)

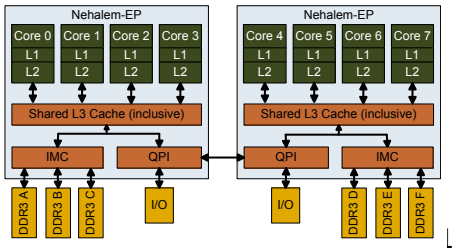
- **Memória partilhada**
  - processadores partilham um barramento de acesso à memória
  - As *caches* reduzem o tráfego no barramento e a latência dos acessos à memória
  - A largura de banda de acesso à memória é partilhada pelos vários processadores => limitação à escalabilidade deste tipo de arquitetura
  - Um valor pode estar replicado em vários sítios => são necessários mecanismos para assegurar a coesão entre as *caches* dos vários processadores e a memória



- **Memória distribuída**
  - Cada processador contém a sua própria memória, existindo uma rede de interligação entre os processadores
  - A largura de banda da memória é proporcional ao número de processadores



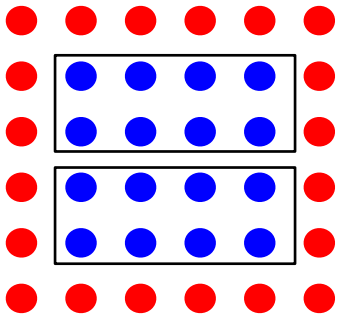
- **Sistemas híbridos**
  - Acesso não uniforme à memória (NUMA)



Paradigmas de Computação Paralela

Programação com memória partilhada VS memória distribuída

- **Memória partilhada (fios de execução)**
  - Matriz é A é partilhada pelos fios de execução
  - Tarefa 1: for(i=0; i<N/2; i++) A[i,j] = ...
  - Tarefa 2: for(i=N/2; i<N; i++) A[i,j] = ...



- **Memória distribuída (processos)**
  - Matriz é A é distribuída pelos processos
  - Tarefa 1:
    Recebe A1
    for(i=0; i<N/2; i++) A1[i,j] = ...
    Envia A1
  - Tarefa 2: Recebe A2; for(i=0; i<N/2; i++) A2[i,j] = ...

Paradigmas de Computação Paralela

Como implementar um paradigma de computação paralela?

- **Estender uma linguagem existente**
  - Biblioteca
    mpi_send(...,...)
  - Diretivas
    #omp parallel for
    for(i=0; i<N; i++) ...
  - Extensão da sintaxe
    - Novo compilador?

- **Desenvolver um nova linguagem**
  - Abordagem mais “limpa”
  - Aceitação mais difícil

```
cilk int fib (int n) {
   if (n < 2) return 1;
   else {
      int rst = 0;
      rst += spawn fib (n-1);
      rst += spawn fib (n-2);
      sync;
      return rst;
   }
}
```

Parallel Computing Paradigms

A methodology to develop parallel applications