Overview of Parallel Programming Models

Silvio Stanzani , Raphael Cóbe , Rogério Iope

UNESP - Núcleo de Computação Científica

silvio@ncc.unesp.br , rmcobe@ncc.unesp.br , rogerio@ncc.unesp.br

Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- Memory System and Vector Processing Units
- Hybrid Parallel Architectures

Agenda

- **Parallelism and Concurrency**
- Parallel Architectures Types
- Memory System and Vector Processing Units
- Hybrid Parallel Architectures

Parallel Processing

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem

- Parallel processing includes techniques and technologies that make it possible to compute in parallel

  - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …

- Parallel computing is an evolution of serial computing
  - Parallelism is natural
  - Computing problems differ in level / type of parallelism

Concurrency

- Consider multiple tasks to be executed in a computer
  - Tasks are concurrent with they can execute at the same time (concurrent execution)

- If a task requires results produced by other tasks in order to execute correctly, the task's execution is dependent
  - Some form of synchronization must be used to enforce (satisfy) dependencies

Parallelism

- Parallelism = concurrency + “parallel” hardware
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware

- Motivations for parallelism
  - Faster time to solution (response time)
  - Solve bigger computing problems (in same time)
  - Effective use of machine resources
  - Overcoming memory constraints

- Serial machines have inherent limitations
  - Processor speed, memory bottlenecks, …

Agenda

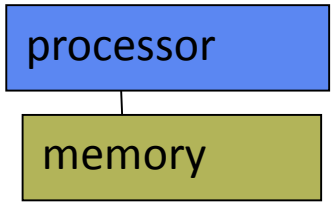
- Parallelism and Concurrency
- **Parallel Architectures Types**
- Memory System and Vector Processing Units
- Hybrid Parallel Architectures

Classifying Parallel Systems Flynn's Taxonomy

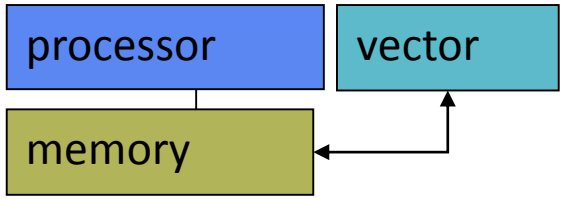
- Distinguishes multi-processor computer architectures along the two independent dimensions
  - Instruction and Data
  - Each dimension can have one state: Single or Multiple
- SISD: Single Instruction, Single Data
  - Serial (non-parallel) machine
- SIMD: Single Instruction, Multiple Data
  - Processor arrays and vector machines
- MISD: Multiple Instruction, Single Data (unusual)
- MIMD: Multiple Instruction, Multiple Data
  - Most common parallel computer systems

Parallel Architecture Types

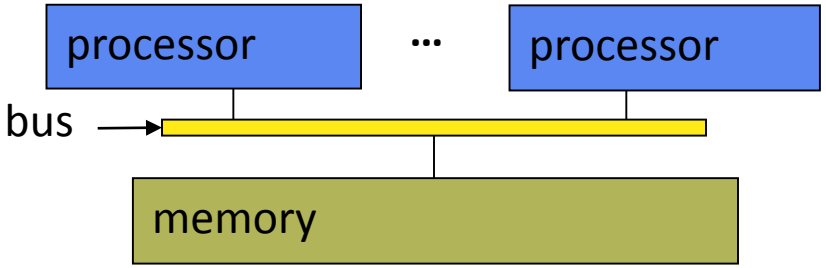
- Uniprocessor
  - Scalar processor



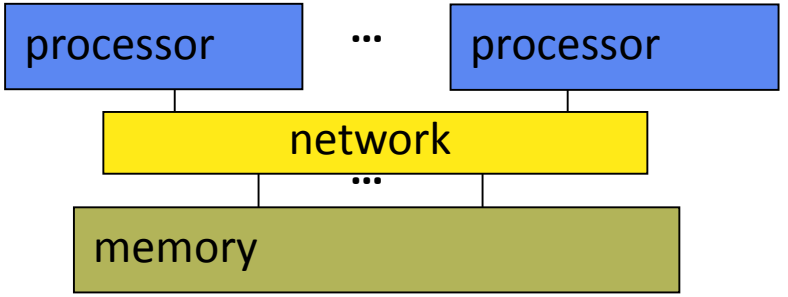
  - Vector processor



- Shared Memory Multiprocessor (SMP)
  - Shared memory address space
  - Bus-based memory system

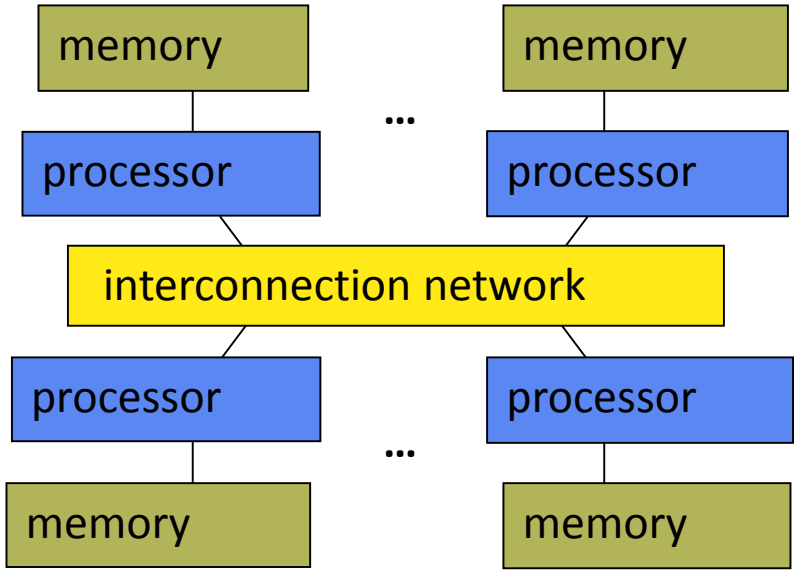


  - Interconnection network



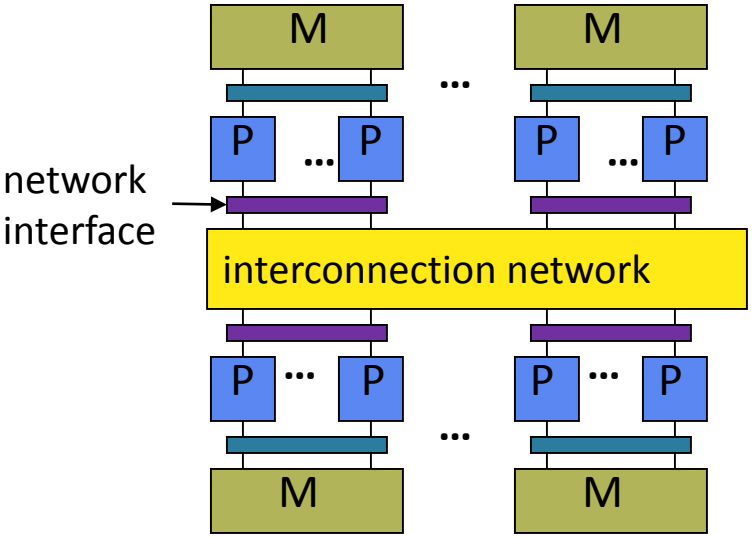
Parallel Architecture Types (2)

- Distributed Memory Multiprocessor
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - Many, many processors

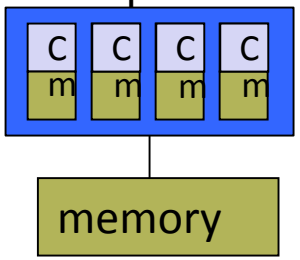
- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



  - Can also be regarded as MPP if processor number is large

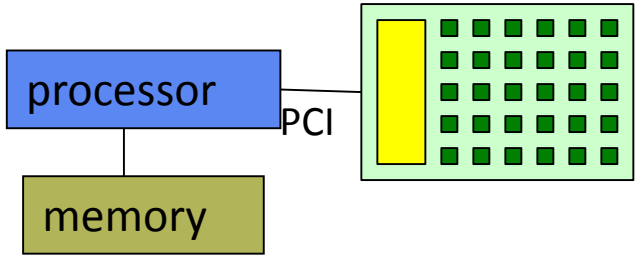
Parallel Architecture Types (3)

- Multicore
  - Multicore processor

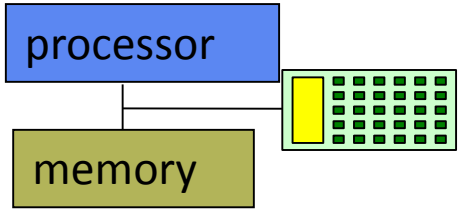


cores can be hardware multithreaded (hyperthread)

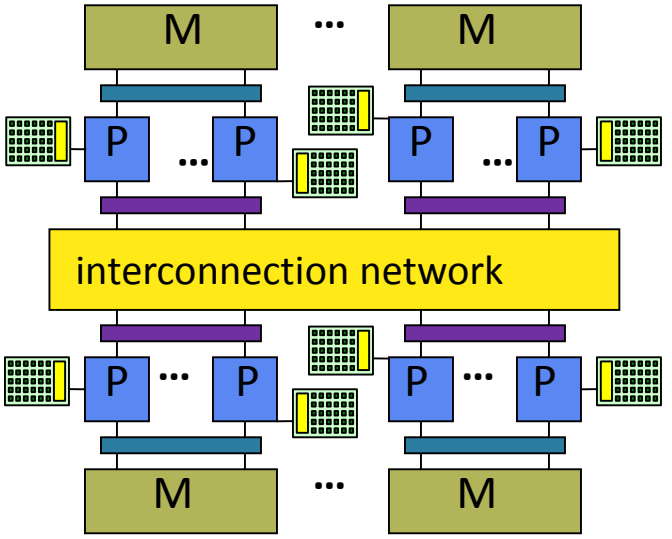
  - Coprocessor



  - "Fused" processor accelerator



- Multicore SMP+coprocessor Cluster
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes
  - Coprocessor attached



Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- **Memory System and Vector Processing Units**
- Hybrid Parallel Architectures

Memory System

Fast

CPU Register: internal Processor Memory. Stores data or instruction to be executed

Cache: stores segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations

Main memory: only program and data currently needed by the processor resides in main memory

Auxiliary memory: devices that provides backup storage

Larger in Size

Cache Memory

- Cache Memory is employed in computer systems to compensate for the difference in speed between main memory access time and processor.

- Operating System controls the load of Data to Cache;
  - such load can be guided by the developer

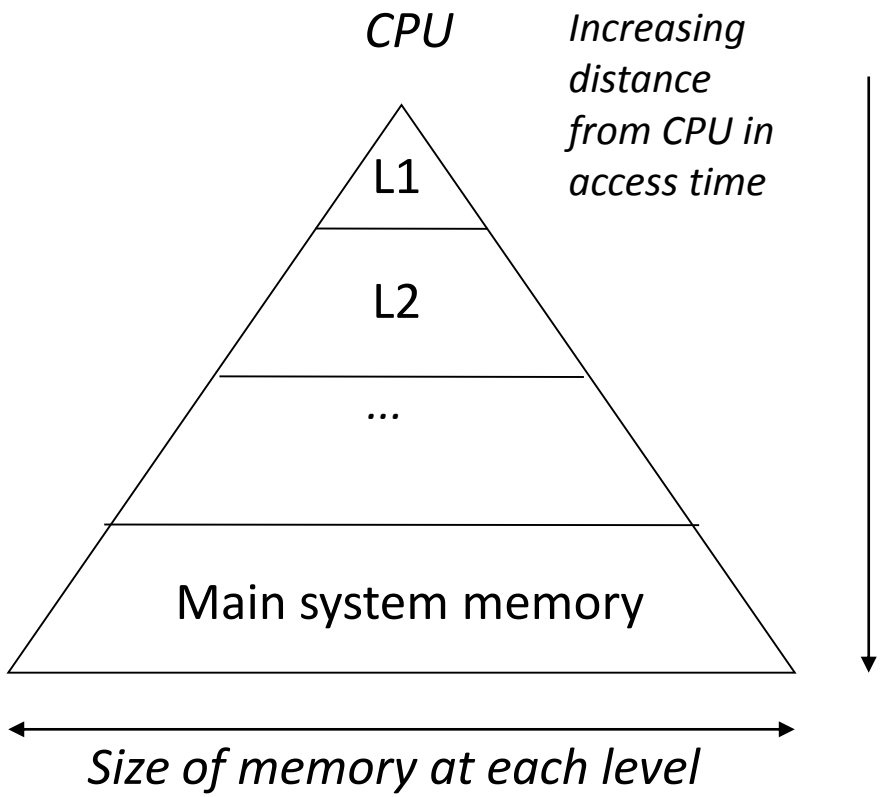
- The performance of cache memory is frequently measured in terms of hit ratio.

  - When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**.

  - If the word is not found in cache, it is in main memory and it counts as a **miss**

Locality

- Temporal locality: if an item was referenced, it will be referenced again soon (e.g. cyclical execution in loops);
- Spatial locality: if an item was referenced, items close to it will be referenced too (the very nature of every program – serial stream of instructions)

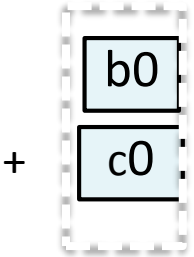


Scalar and Vector Instructions

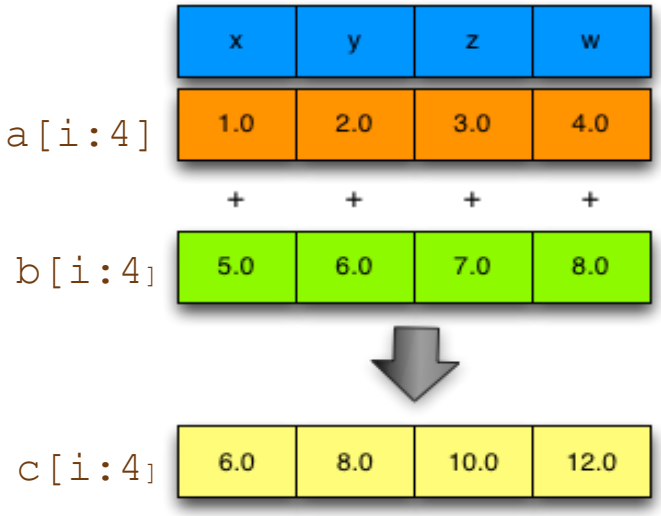
- **Scalar** Code computes this one-element at a time.
- **Vector (or SIMD)** Code computes more than one element at a time.
  - SIMD stands for **S**ingle **I**nstruction **M**ultiple **D**ata.
- Vectorization
  - Loading data into cache accordingly;
  - Store elements on SIMD registers or vectors;
  - Iterations need to be independent;
  - Usually on inner loops.

```
float *A, *B, *C;
for(i=0;i<n;i++){
  A[i] = B[i] + C[i];
}
```

- Scalar



- SIMD



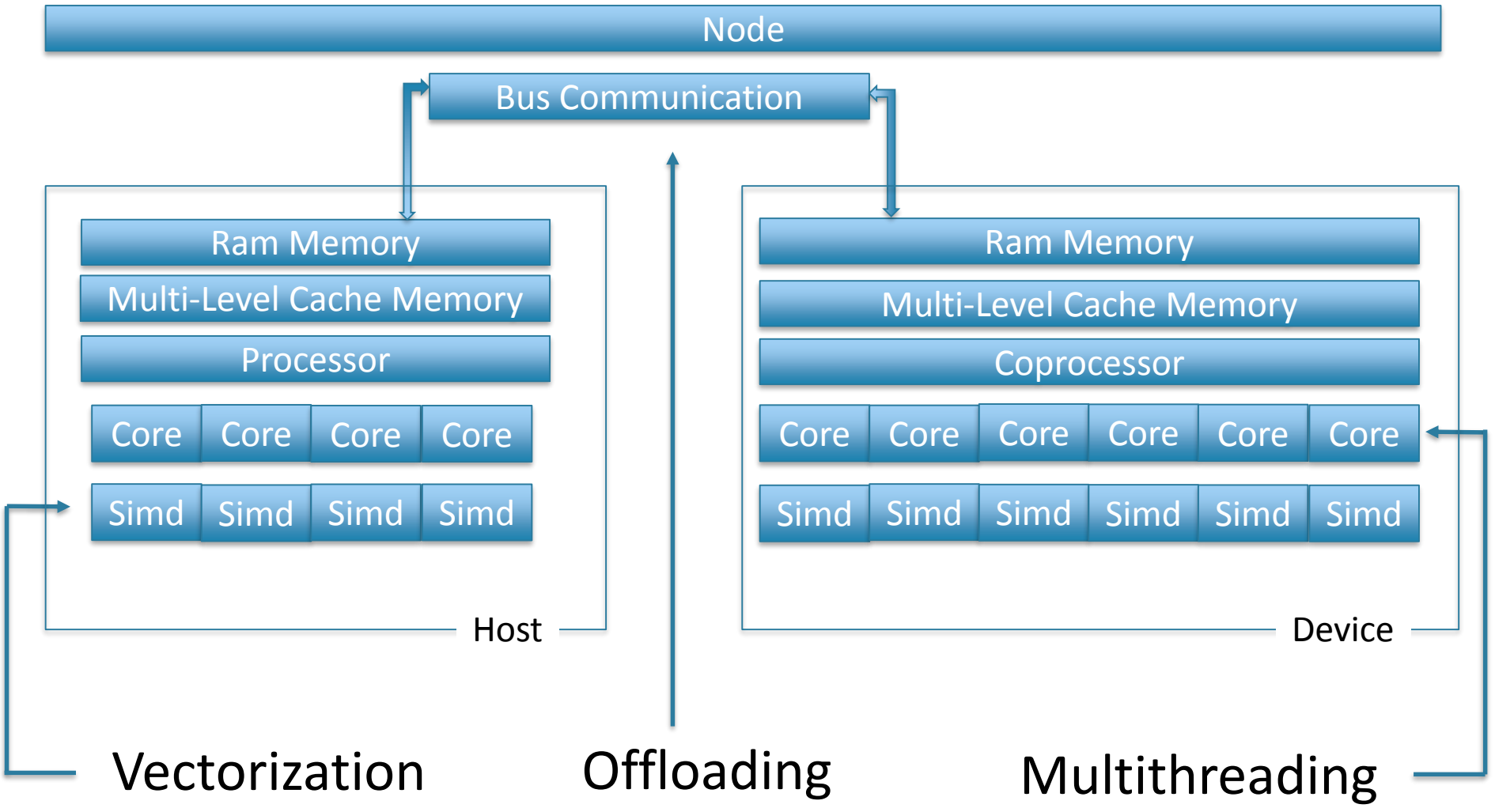
Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- Memory System and Vector Processing Units
- **Hybrid Parallel Architectures**

Hybrid Parallel Architectures

- Heterogeneous computational systems:
  - Multicore processors;
  - Multi-level memory sub-system;
- Multi-level parallelism:
  - Processing core;
  - Chip multiprocessor;
  - Computing node;
  - Computing cluster;
- Hybrid Parallel architectures
  - Coprocessors and accelerators;

Hybrid Parallel Architectures



Hybrid Parallel Architectures

- Exploring parallelism in hybrid parallel architectures
  - Multiprocessing
  - Multithreading
  - Vectorization
    - ❑ Auto vectorization
    - ❑ Semi-auto vectorization
    - ❑ Explicit vectorization
  - Offloading
    - ❑ Offloading code to device