Unit 1: Why parallel computing?

Video lesson 1: serial vs. parallel

Eduard Ayguadé, Josep Ramon Herrero,
Daniel Jiménez and Gladys Utrera

Barcelona Supercomputing Center
Universitat Politècnica de Catalunya

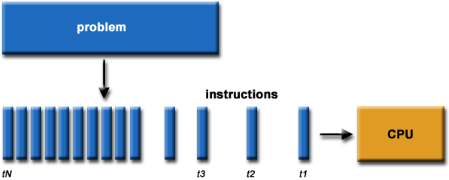
UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Serial execution

Usually programs are written with a serial execution model in mind

- Program is composed of a sequence of instructions (arithmetic, memory read and write, control, ...) ...
- ... to be run on a computer with a single processor (CPU)



- Instructions are executed one after another, only one at any moment in time

Serial execution

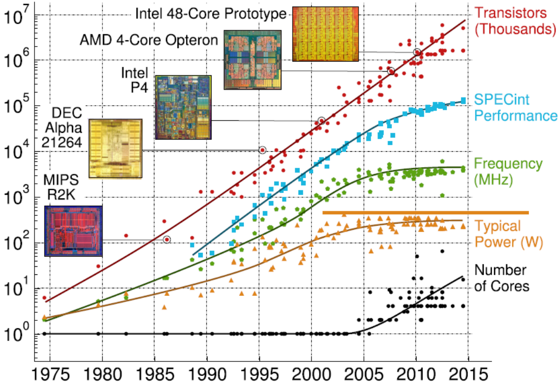
The execution time of a program with N instructions on a processor that is able to execute F instructions per second is

$$T = N \div F$$

One could execute the program faster (i.e. reduce T) by augmenting the value of F. And this has been the trend during more than 30 years of technology and computer architecture evolution.

Uniprocessor and multicore performance evolution[1]

Tecnological limitations lead microprocessor industry to multicores



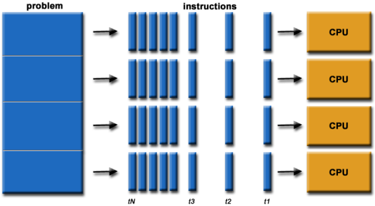
[1] Data collected by M.Horowitz et al.

Parallel execution

So another way to reduce the execution time of a program T

$$T = N \div F$$

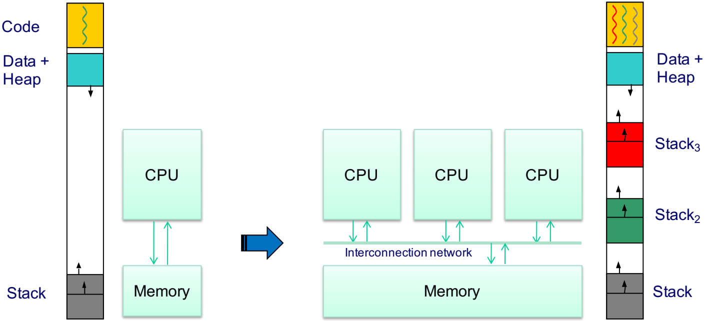
would be to split the program into discrete parts, to be called tasks, and use multiple processors (CPUs) to execute them at the same time



And data?

Parallel architectures (1)

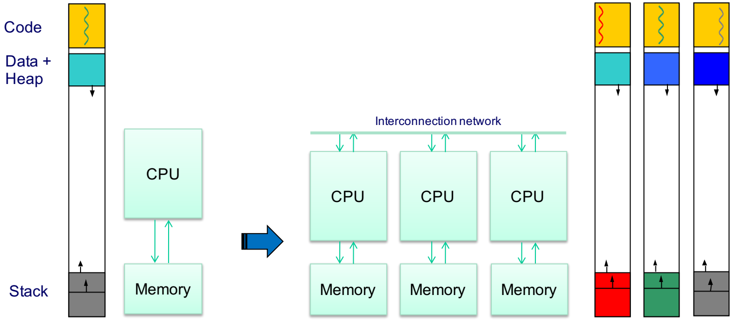
Shared-memory architecture and memory address space



Hardware support for coherent data sharing and tight synchronization

Parallel architectures (2)

Distributed-memory architecture and memory address space



Hardware support for remote data accesses and communication

Parallel execution

Ideally, each processor could receive $\frac{1}{P}$ of the program, reducing its execution time by P

$$T = (N \div P) \div F$$

Need to manage and coordinate the execution of tasks, ensuring correct access to shared resources

Throughput vs. parallel computing

Throughput computing: multiple, unrelated, instruction streams (programs) executing at the same time on multiple processors

In the simplest way, k programs on P processors; each program receives $\frac{P}{k}$ processors

Unit 1: Why parallel computing?

Video lesson 1: serial vs. parallel

Eduard Ayguadé, Josep Ramon Herrero,
Daniel Jiménez and Gladys Utrera

Barcelona Supercomputing Center
Universitat Politècnica de Catalunya