Foundations

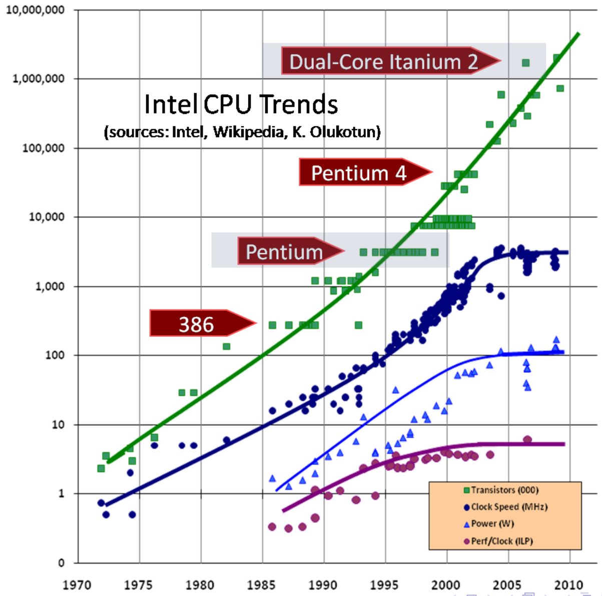
André Cerqueira

FCUP

2024

Este trabalho foi financiado por: Projeto 10110190 – EUROCC2, com apoio financeiro da FCT/MCTES através de fundos nacionais (PIDDAC).

Performance Trend over the Years



Performance Trend over the Years

- In the past, programmers could rely on innovations in hardware, architecture and compilers to double performance of their programs every 2 years without having to change a line of code.
- As of 2006, the manufacturer's solution was to start having multiple processors per chip, generically called multicore processors, where the benefit is often more on throughput than on response time and go for parallel computing.

Why Go Parallel ?

Major Reasons:

- Reduce the execution time
- Be able to solve larger and more complex problems

Other Important Reasons:

- Computing Resources are frequently under-utilized
- Overcome the physical limitations in chip density and production costs of faster sequential computers

Throughput / Latency

Throughput:

- Number of computing tasks per time unit
- ie: 1000 credit card payments in a minute.

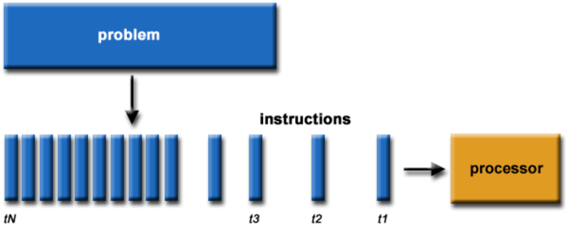
Latency:

- Delay between invoking the operation and getting the response.
- ie: time taken to process a credit card transaction.

When optimizing performance, an improvement in one factor (such as throughput) may lead to the worsening in another factor (such as latency).

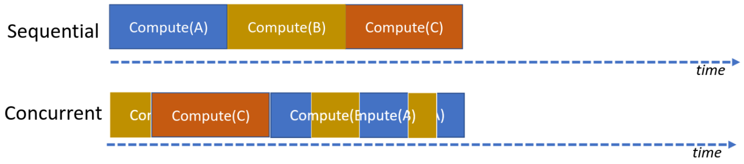
Sequential Computing

Sequential computing occurs when a problem is solved by executing one flow of instructions in one processing unit.



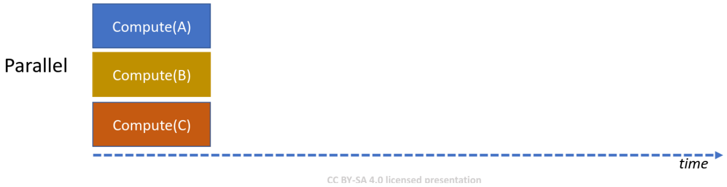
Concurrency

A program exhibits concurrency (or potential parallelism) when it includes tasks (contiguous parts of the program) that can be executed in any order without changing the expected result.



Parallel Computing

Parallel computing occurs when a problem is decomposed in multiple parts that can be solved concurrently
Each part is still solved by executing one flow of instructions in one processing unit but, as a whole, the problem can be solved by executing multiple flows simultaneously using several processing units..



Implicit Parallelism

Parallelism is exploited implicitly when it is the compiler and the runtime system that:

- Automatically detect potential parallelism in the program
- Assign the tasks for parallel execution
- Control and synchronize execution

Advantages and disadvantages:

- (+) Frees the programmer from the details of parallel execution
- (+) More general and flexible solution
- (–) Very hard to achieve an efficient solution for specific problem

Explicit Parallelism

Parallelism is exploited explicitly when it is left to the programmer to:

- Annotate the tasks for parallel execution
- Assign tasks to the processing units
- Control the execution and the synchronization points

Advantages and disadvantages:

- (+) Experienced programmers achieve very efficient solutions for specific problems
- (–) Programmers are responsible for all details of execution
- (–) Programmers must have deep knowledge of the computer architecture to achieve maximum performance

Heterogeneous Systems

- Heterogeneous computing involves the integration of various types of processors within a single computational framework.
- It enables tasks to be allocated to processors best suited for them, enhancing performance and expanding application possibilities.

Heterogeneous Systems

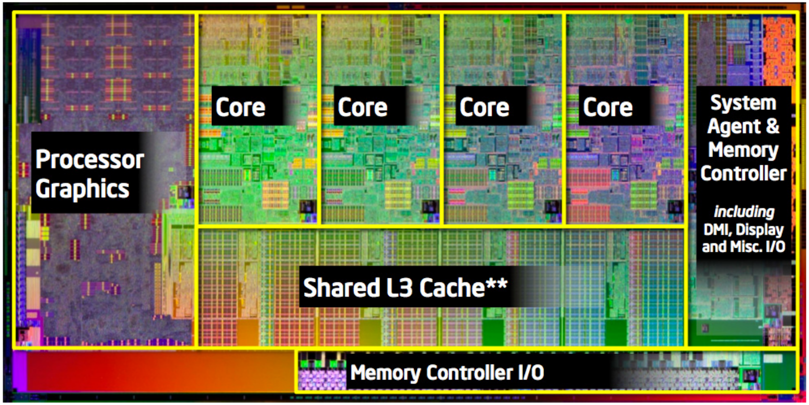


Figure: CPU + GPU heterogeneous system

- Uses various processor types within one framework.
- Allocates tasks to the most suitable processor.

Heterogeneous Systems

Challenges:

- Distributing workloads across multiple processors has been complex.
- Managing different processor types add complexity and requires additional effort.

Heterogeneous Systems

Challenges:

- Distributing workloads across multiple processors has been complex.
- Managing different processor types add complexity and requires additional effort.

Solutions:

- Streamlined Programming: Language that simplifies workload distribution.
- Unified Frameworks: Development of frameworks that abstract the complexity of heterogeneous architectures, easing adoption.

Heterogeneous Systems

Challenges:

- Distributing workloads across multiple processors has been complex.
- Managing different processor types add complexity and requires additional effort.

Solutions:

- Streamlined Programming: Language that simplifies workload distribution.
- Unified Frameworks: Development of frameworks that abstract the complexity of heterogeneous architectures, easing adoption.

Overcoming these barriers paves the way for broader adoption of heterogeneous computing, unlocking its potential for various applications, including high performance computing (HPC).

And that's why Sycl was created!