Arquitecturas Paralelas de Computadoras

Introducción

Profr. Carlos Ernesto Carrillo Arellano[1]

[1]Universidad Autónoma Metropolitana - Unidad Azcapotzalco
Departamento de Ingeniería Electrónica
Correo electrónico: ceca@xanum.uam.mx

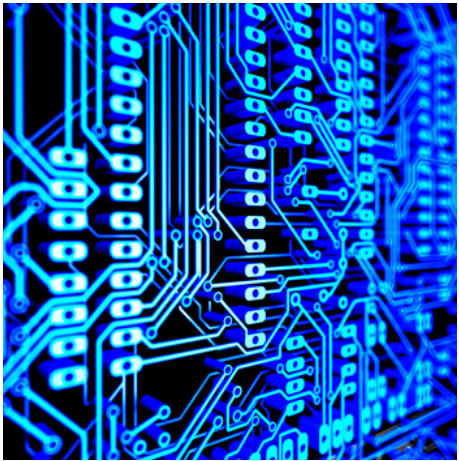
Mayo, 2019

Contenido

Contenido

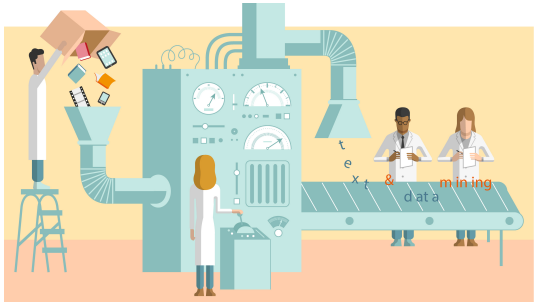
Introduction

1. Throughout the history of computer systems, parallel computers have been an **important class of computers**.
2. A parallel computer **combines a large number of processing elements (CPUs) into a single system**, allowing a large computation to be carried out in orders of magnitude **shorter execution time**.
3. Scientists and engineers have relied on parallel computers to **solve important scientific questions** by running simulations on them.



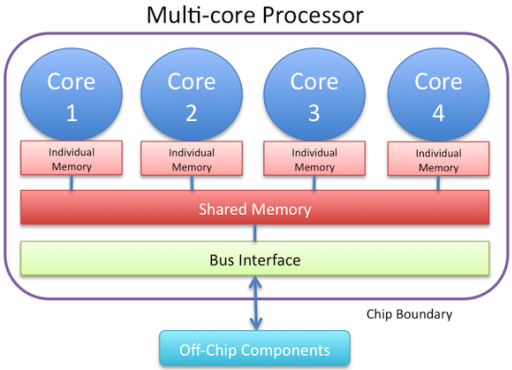
Introduction

1. Parallel computers have found a **broader audience.**
2. Corporations rely on **mining data** from a large collection of databases, which is a very computation intensive process.
3. Businesses rely on **processing transactions** on powerful parallel computers, while internet search engine providers use parallel computers to **rank web pages and evaluate their relevance** based on a search criteria.
4. Gamers demand games that show **more realistic simulation** of physical phenomena and realistic rendering of three-dimensional images.

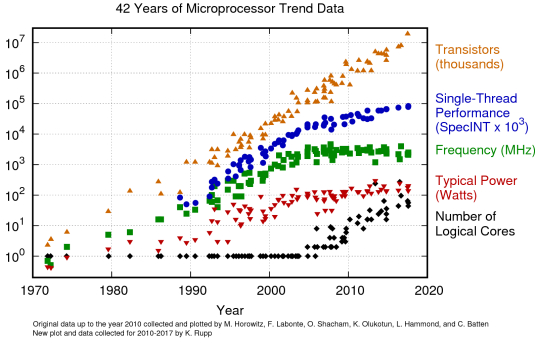


Introduction

1. The **parallel computer architectures** evolved from the architecture of big, powerful, and expensive computer systems to the mainstream architecture of servers, desktops, and embedded systems.
2. Before 2001, parallel computers were mainly used in the **server** and **super computer** markets. **Client machines** (desktops, laptops, and mobile devices) were **single-processor systems.**
3. Since 2001 they have been evolving into an architecture in which **multiple processor cores** are implemented in a **single chip.**
4. Such an architecture is popu larly known as the **multicore architecture.**



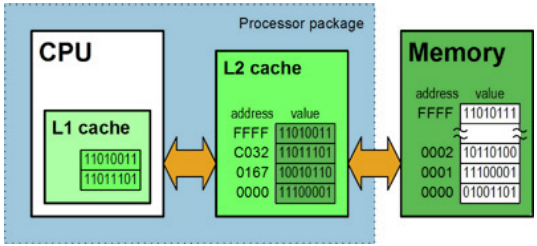
Introduction



1. An enabling trends for the move to multicore architectures has been the **increasing miniaturization of transistors**, through which more and more transistors can be packed in a single chip.
2. **Moore's Law** (1965): the number of transistors that can be manufactured inexpensively in a single Integrated Circuit (IC) doubles every two years.
3. More and more components of a single processor on a chip, followed by a**dding more features to a single processor**, and more recently has resulted in **replicating processor cores in one chip.**

Introduction

1. Transistor integration was used to **move components of a single processor** that did not fit on chip into the chip, e.g., FPU
2. Since the speed of **main memory** was not keeping up with the growth of speed of the processor, it was necessary to introduce **memory hierarchy**, where a smaller memory was integrated on chip for faster data access. (L1 and L2 caches)
3. If **transistor integration continues**, it is likely that the **processor chip will integrate more processor cores, larger caches and/or deeper cache hierarchy, and other components** such as the memory controller and a graphic processor.

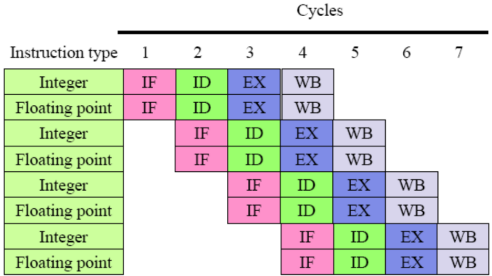


Evolution of parallel computers

1. Parallel architectures were initially a **natural idea** because there were not enough transistors on a chip to implement a complete microprocessor.
2. It was natural to **have multiple chips** that **communicated with each other**, either when those chips implemented different components of a processor or when they implemented components of different processors.
3. All **levels of parallelism** were considered in parallel computer architectures: **instruction level parallelism, data parallelism, and task parallelism.**

Evolution of parallel computers

1. Over time, it became clear which types of parallelism were more appropriate for implementation **across processors or within a single processor.**
2. **Instruction level parallelism** is now implemented in the architecture of a single processor
3. **It requires register-level communication among instructions,** which can only be supported with a low latency on a single processor.
4. A **superscalar processor** is a CPU that implements a form of parallelism called **instruction-level parallelism** within a singleprocessor.

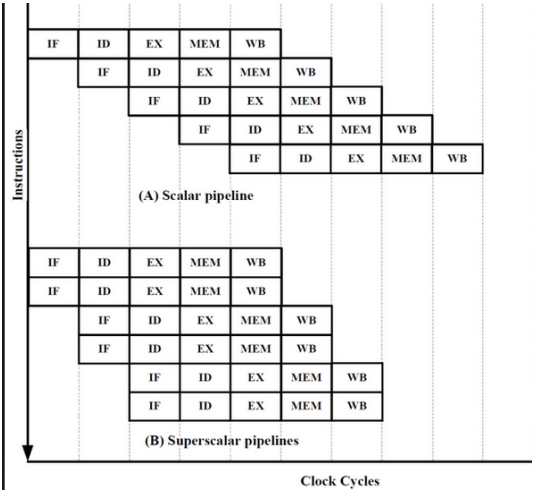


Evolution of parallel computers

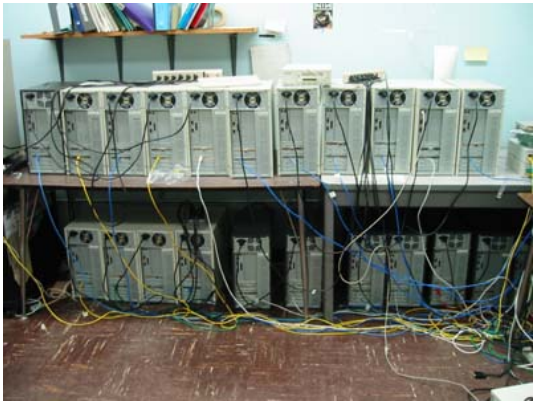
1. As transistor integration continued, an entire microprocessor fit into a single chip (e.g., Intel4004 in 1971).
2. A rapid performance growth was both fueled by transistor miniaturization which drove up the clock frequency of the processor, as well as two classes of architecture techniques:**instruction-level parallelism (ILP) and cache memory.**
3. Gains from these architecture techniques were so significant that uniprocessor systems could catch up the performance of parallel computers in a few years, while costing a tiny fraction of parallel computers.

Instruction level parallelism

1. Instruction-level parallelism refers to the execution of **multiple instructions in parallel**.
2. A superscalar architecture which widens the pipeline so that **multiple independent instructions can be processed simultaneously at the same pipeline stage**, gave further boost to uniprocessor system performance.
3. In addition, **out-of-order execution helped to improve the performance of superscalar architectures** by letting younger (independent) instructions to execute even though older instructions stall in the pipeline.

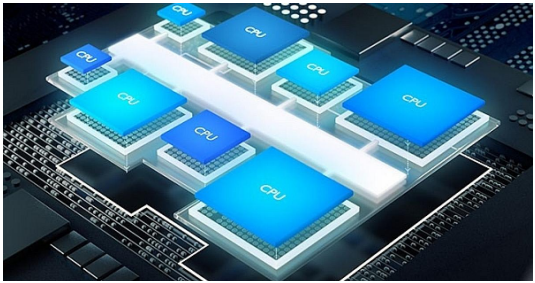


Evolution of parallel computers



1. Low-cost distributed computers were being developed in the 90s by **assembling many uniprocessor systems** with an off-the-shelf network.
2. This gave birth to network of workstations, which was later more commonly referred to as **clusters**.
3. Compared to parallel computers, distributed computers were a lot cheaper, but had a high communication latency between processors.
4. Some classes of applications did not have much inter-processor communication, and were quite scalable when they ran on clusters.

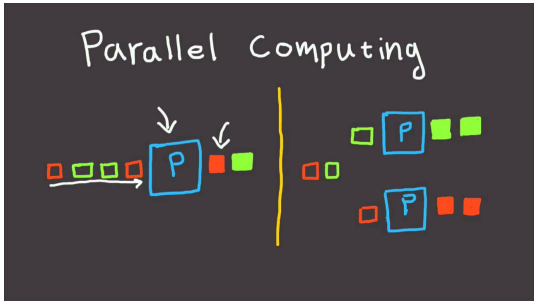
Evolution of parallel computers



1. Chipmakers had no choice but to abandon the ILP in favor of implementing multiple processors on a chip. This new architecture is referred to as Chip Multi-Processors (CMP) or more popularly **multicore architectures**.
2. The trend continued towards more and **more processors being integrated in a single chip.**
3. This makes **parallel computers ubiquitous** in the servers, desktops, and even mobile systems.
4. Some chips adopt the approach of having **fewer powerful processor cores** such as Intel Core Duo, while others adopt the approach of having **many simple processor cores**, such as Sun Niagara.

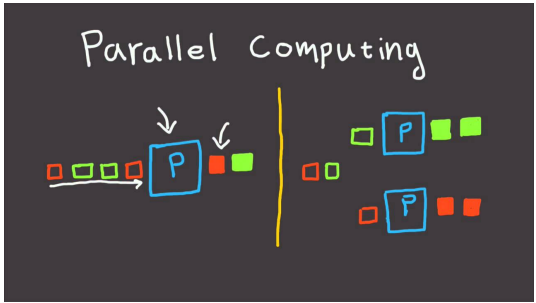
Parallel computers

1. "**A parallel computer is a collection of processing elements that communicate and cooperate to solve a large problem fast**."[Almasi and Gottlieb, 1989]
2. More recently, the de-facto **processing** elements are **processors**, hence parallel computers are also referred to as multiprocessors; multicore is more specific term, it refers specifically to **multiple processors implemented on a single chip**.
3. The term **communicate** refers to the **processing elements sending data to each other**.
4. The choice of communication mechanisms determine two important classes of parallel architectures: **shared memory systems** or **message passing systems**.



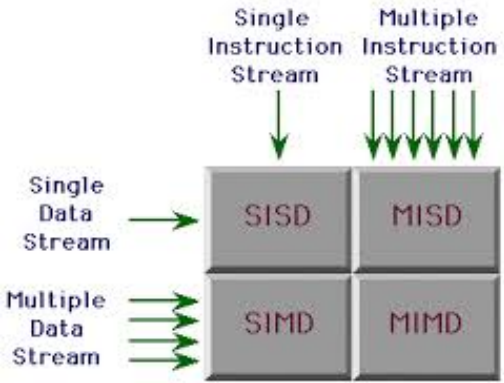
Parallel computers

1. "**A parallel computer is a collection of processing elements that communicate and cooperate to solve a large problem fast.**"[Almasi and Gottlieb, 1989]
2. The term **cooperate** refers to the **synchronization of the progress of execution** of a parallel task relative to other tasks.
3. Synchronization allows **sequencing of operations** to ensure correctness.



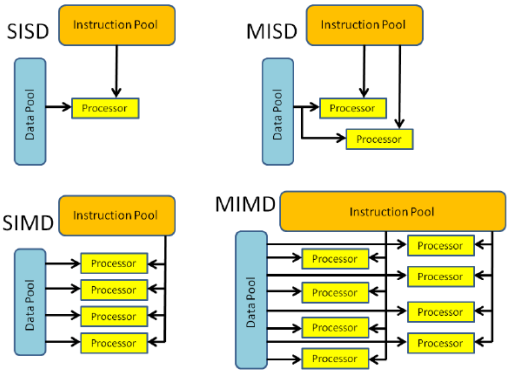
Flynn's Taxonomy

1. Flynn defined a **taxonomy** of parallel computers [Flynn, 1972] based on the **number of instruction streams and data streams**
2. An **instruction stream** is a sequence of instructions followed from a single program counter.
3. A **data stream** is an address in memory which the instruction operates on.
4. A control unit fetches instructions from a single program counter, decodes them, and issues them to the processing element. Instruction and data are both supplied from the memory.



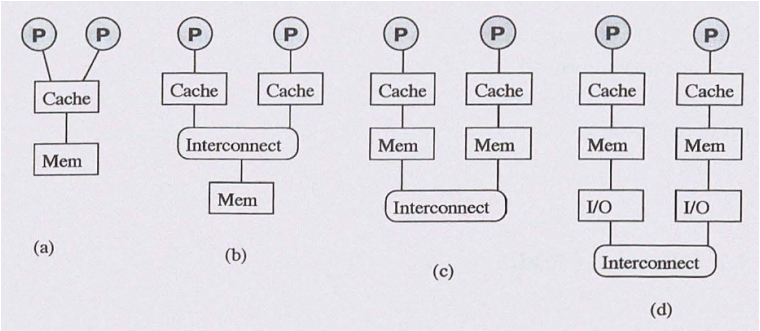
Flynn's Taxonomy

1. **SISD** is not considered a parallel architecture as it only has one instruction and one data stream. SISD exploits parallelism at the instruction level.
2. **SIMD** is a parallel architecture in which a single instruction operates on multiple data. In a SIMD architecture, only one instruction is needed to operate on a large data.
3. **MISD** is an architecture in which multiple processing relements execute from different instruction streams, and data is passed from one processing element to the next.
4. **MIMD** is the architecture used in most parallel computers today. It is the most flexible architecture since there is no restriction on the number of instruction streams or data streams,



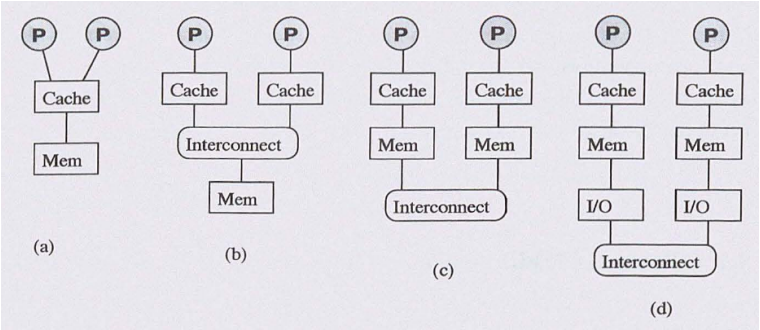
MIMD Parallel Computers

1. ¿At which level processors in a MIMD architecture are physically interconnected?
2. a) Share the cache, b) Interconnect private caches
3. b) Because all processors can access the memory with the same latency, then this class of MIMD computers are also referred to as **uniform memory access** (UMA) architectures.



MIMD Parallel Computers

1. c) Each processor has its own caches and local memory, interconnect across all local memories to give the abstraction of a single memory. However, the memory access latency varies since remote memory takes longer to access than local memory. Such architectures are referred to as **non-uniform memory access** (NUMA).
2. Each processor is a complete node with its own caches, local memory, and disk; and the interconnection is provided at the level of I/0 connection.



MIMD UMA parallel computers

1. Processors of Bus-based multiprocessors that experience the **same uniform access time to any memory module in the system** are often referred to as Uniform Memory Access (UMA) systems or Symmetric Multi-Processors (SMPs).
2. With UMA systems, the CPUs are connected via a system bus (Front-Side Bus) to the **Northbridge**. The **Northbridge** contains the memory controller and **all communication to and from memory must pass through the Northbridge**.
3. The I/O controller, responsible for managing I/O to all devices, is connected to the Northbridge. Therefore, **every I/O has to go through the Northbridge to reach the CPU**.