How Multicore Processors Work and Intel's P-Core & E-Core Architecture

Modern computing requires processors to handle increasingly complex workloads efficiently. Early computers used **single-core processors**, meaning they had only one processing unit to execute instructions sequentially. While single-core processors execute instructions sequentially, they can still perform multitasking via time-sharing. The CPU rapidly switches between tasks, creating the illusion of simultaneous execution, but not true parallel execution..

To improve performance, **multicore processors** were developed, integrating multiple processing units (**cores**) into a single chip. This allows for **parallel execution of instructions**, significantly enhancing multitasking, responsiveness, and efficiency.

Intel introduced its hybrid multicore architecture with the Lakefield processors in 2020 (though it was announced in 2019), however it became mainstream with the 12th-generation Alder Lake processors (2021), further refining how modern CPUs handle workloads. This design includes two types of cores: **Performance Cores (P-Cores)** for high-performance tasks and **Efficiency Cores (E-Cores)** for power-efficient background processing. This hybrid model optimizes workload distribution, balancing performance and power consumption.

How Multicore Processors Work

A **multicore processor** consists of multiple independent **cores**, each capable of executing its own set of instructions. This allows for **parallel processing**, where multiple tasks can be executed simultaneously. Unlike single-core processors, which execute instructions sequentially, multicore processors can divide workloads among different cores, reducing execution time and increasing performance.

Parallel Processing and Task Scheduling

In a multicore processor, the **operating system (OS)** is responsible for distributing tasks among the available cores. This is managed by the **task scheduler**, which assigns processes and threads to different cores based on workload demands.

For example, if a user is running a web browser, a media player, and email, the OS may allocate each of these tasks to separate cores to maximize efficiency. In contrast, on a single-core processor, the CPU would need to **rapidly switch between these tasks**.

Applications designed for **multi-threading** can further enhance performance by dividing tasks into **multiple threads** that can be executed simultaneously on different cores. For instance, in video editing software, different cores can be assigned to process separate video frames in parallel, significantly reducing rendering times.

However, efficient task allocation isn't automatic. Some applications are not optimized for multi-threading and rely heavily on **single-threaded performance**, meaning they perform better with a few high-speed cores rather than a large number of slower cores.

Challenges of Multicore Processing

Despite the advantages, multicore processors introduce new challenges. The OS must efficiently **schedule tasks** across cores to ensure an even workload distribution. Poor scheduling can result in some cores being underutilized while others are overloaded, reducing efficiency.

Additionally, **inter-core communication latency** can impact performance. When multiple cores need to share data, it must be transferred through cache memory or system memory, which introduces slight delays. The **cache hierarchy**, described below, plays a crucial role in minimizing this issue.

Role of Cache Memory in Multicore Processors

Cache memory is a small, high-speed memory located inside the processor, designed to store frequently accessed data and instructions. Since accessing data from RAM (Random Access Memory) is significantly slower, cache memory helps reduce delays by keeping frequently used data close to the processor.

The cache system is organized in multiple levels: **L1, L2, and L3**, each serving a distinct role in optimizing performance.

L1 Cache (Level 1)

The **L1 cache** is the fastest and closest to the processor core. It is typically split into two sections:

1. **Instruction Cache (L1i)** – Stores frequently used CPU instructions.
2. **Data Cache (L1d)** – Stores recently accessed data.

Each core has its own dedicated L1 cache, ensuring the fastest possible access to instructions and data. However, due to its high speed, L1 cache is very small, typically ranging from **32KB to 128KB per core**.

L2 Cache (Level 2)

The **L2 cache** is larger than L1 but slightly slower. It serves as a secondary buffer, storing data that could not fit into L1. L2 cache may be **dedicated per core** or shared between a few cores.

L2 cache sizes typically range from **256KB to 2MB per core**, depending on the processor architecture. It helps reduce the frequency of data retrieval from the slower L3 cache or RAM.

L3 Cache (Level 3)

The **L3 cache** is the largest and slowest of the three levels but plays a crucial role in **shared data access**. Unlike L1 and L2, which may be private to individual cores, the L3 cache is usually **shared among all cores** in a processor.

L3 cache acts as a common data repository, ensuring that frequently used data is available to multiple cores without needing to retrieve it from RAM. L3 cache sizes vary widely, typically ranging from **4MB to 64MB**, depending on the processor model.

If data is not found in any cache level, the processor must fetch it from RAM, which introduces significant delays.

Intel’s Hybrid Architecture: P-Cores and E-Cores

To further enhance performance and power efficiency, Intel introduced a hybrid architecture that combines high-performance cores (P-Cores) and power-efficient cores (E-Cores). This allows the processor to dynamically balance performance-heavy tasks and background tasks.

Performance Cores (P-Cores)

P-Cores are optimized for single-threaded performance and high-speed processing. These cores feature out-of-order execution, allowing them to dynamically rearrange instruction sequences for faster execution. This improves performance in applications such as gaming, software development, and 3D rendering.

Some P-Cores support hyper-threading (HT), meaning each core can execute two threads simultaneously. However, not all Intel processors enable Hyper-Threading on P-Cores. Due to their higher clock speeds and complex architecture, P-Cores consume more power and generate more heat.

Efficiency Cores (E-Cores)

E-Cores are optimized for multi-threaded workloads and power efficiency. Unlike P-Cores, E-Cores do not support hyper-threading, meaning each core handles only one thread at a time. However, they are designed to be highly efficient, allowing multiple E-Cores to work together effectively.

E-Cores are ideal for handling background processes, such as system updates, media playback, and low-priority tasks. By offloading these tasks from P-Cores, the system optimizes power consumption without compromising performance.

In laptops and mobile devices, E-Cores significantly improve battery life by reducing power usage during low-demand activities.

Workload Management: Intel Thread Director

To effectively manage workload distribution between P-Cores and E-Cores, Intel introduced **Thread Director**, a hardware-based scheduling system. Thread Director works in real-time with the **operating system (OS)** to assign tasks to the most appropriate core type.

When a high-performance task is detected, such as **gaming or video rendering**, Thread Director prioritizes P-Cores. If background tasks, such as **file downloads or antivirus scans**, are running, they are assigned to E-Cores to prevent performance degradation.

By dynamically optimizing task allocation, Thread Director ensures that **high-priority applications receive the necessary processing power while background tasks run efficiently**. This improves overall system responsiveness and power efficiency.