Chapter 2
Computer Architectures and Programming Models:
How to Exploit Parallelism

ABSTRACT

In this chapter, basic concepts about programming models and computer architectures are introduced to provide context about the major developments in both topics. Differences between multicore and accelerators are also addressed to help the reader understand how the concepts relate and translate between different architectures. Moreover, the authors also present an introduction to programming models focusing on their suitability for different parallelizing strategies and their special features. In this way, the reader is guided to identify which programming models are best suited for specific problems and architectures, according to the computational requirements, as well as those arisen from the data layout.

ARCHITECTURES

In 1965, Gordon Moore, Intel's co-founder noticed that advances in technology were moving so fast that it was possible to enhance processors in such a way that the number of transistors per chip would double every 18 months. This trend was stated as "Moore's Law", and it continues almost strictly until today (Edwards, 2021). This increase of resources has been mainly aimed at sequential processors following a common computer model known as Von

DOI: 10.4018/978-1-7998-7082-1.ch002

Neumann architecture. In this way, it is not necessary to retrain programmers or rewrite programs for emergent computer architectures. Multiple transparent techniques, from a programmer point of view, were carried out to take advantage of a higher number of transistors. However, the use of these techniques and the limitation of the current VLSI (Very Large Scale Integration) technology do not suppose an increase in performance for single-threaded processors today. In response, most hardware companies are designing and developing new parallel architectures (Geer, 2005). In order to maintain the same rate of growth in performance as in the number of transistors per chip, the processor designer evolved from the Von Neumann model to a new concept of Chip Multi-Processors (CMP), which integrates several processing elements in the same chip. This change supposes a greater pressure on memory and I/O buses as the number of cores increases. Furthermore, this new paradigm implies higher efforts for programmers who are responsible for exposing the implicit parallelism in their applications.

Computer architectures have suffered many changes over time. These were carried out to search for more amenable features to solve scientific problems. In this way, some proposals for massively parallel computers emerged in the 1950s (Unger, 1958) because of the fact that, in most of these applications, it was necessary a high number of identical calculations, so that it seemed logical to replicate the processing elements. This led to the first Simple Instruction Multiple Data (SIMD) architectures; ILLIAC IV (Barnes, et al., 1968), MPP (Batcher, 1980), CLIP4 (Duff, 1976), and Multiple Instruction Multiple Data (MIMD) architectures; Cytocomputer (Loughead & Cubbrey, 1980), ZMOB (Rieger et al., 1981).

Numerical simulation is probably the field with the most impact on computer architecture. Indeed, those applications related to Linear Algebra are one of the most computationally expensive, mainly due to the huge computational resources needed. Clearly, this has to do with the incorporation of certain SIMD instructions in commercial processors; Intel (Peleg & Weiser, 1996), Sun (Tremblay et al., 1996), HP (Lee, 1996), among others.

Hardware multithreading has a long history as a design strategy for improving aggregate performance on parallel workload. The peripheral processor of the Control Data Corp (CDC 6600) developed in the 1960s and the Heterogeneous Element Processor system developed in the late 1970s (Smith, 1981) are notable examples of the early use of hardware multithreading. Many more multithreading processors have been designed over the years; for