CHAPTER 12: INTRODUCTION TO PARALLEL PROCESSING

Parallel processing is commonly used to improve computer system performance. A uniprocessor system may incorporate parallelism using an instruction pipeline, a fixed or reconfigurable arithmetic pipeline, I/O processors, vectored arithmetic units, and multiport memory.

Multiprocessor systems can be classified using several attributes. Flynn’s classification categorizes multiprocessor systems by their instruction and data streams. Flynn classification, or Flynn’s taxonomy, was proposed by Michael J. Flynn in 1966. The four classifications defined by Flynn are based upon the number of concurrent instruction (or control) and data streams available in the architecture. Single instruction, single data stream, or SISDs, are a sequential computer which exploits no parallelism in either the instruction or data streams. SISD architecture examples include traditional uniprocessor machines like a PC or old mainframes. Single instruction, multiple data streams, or SIMDs, are computers which exploit multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. Examples of SIMDs include array processors or GPUs. Multiple instruction, single data stream, or MISDs, operate on a single data stream. It is an uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples of MISDs include the Space Shuttle flight control computer. Finally, there is multiple instruction, multiple data streams, or MIMDs. MIMDs are multiple autonomous processors simultaneously executing different instructions on different data. Distributed systems are generally recognized to be MIMD architectures; either exploiting a single shared memory space or a distributed memory space. As of 2006, all of the top 10 and most of the TOP500 supercomputers are based on MIMD architecture. MIMD architecture can be further subdivided into the following categories: single program, multiple data (SPMD), where multiple autonomous processors are simultaneously executing the same program (but at independent points, rather than in the lockstep that SIMD imposes) on different data. Also referred to as ‘single process, multiple data’, SPMD is the most common style of parallel programming. Multiple program multiple data, or MPMD, is where multiple autonomous processors simultaneously operate at least two independent programs. Typically such systems pick one node to be the “host” or “manager”, which runs one program that farms out data to all the other nodes which all run a second program.

A system may also be classified by its topology, whether it uses a shared bus, where a set of clients are connected via a shared communications line called a bus, ring topology, where each node connects to exactly two other nodes, forming a single continuous pathway for signals through each node, tree (or hierarchical topology), in which a central ‘root’ node is connected to one or more other nodes that are one level lower in the hierarchy with a point-to-point link between each of the second level nodes and top level central ‘root’ node, mesh, where each node in the network may act as an independent router, regardless of whether it is connected to another network or not, hypercube, which is a multidimensional mesh, completely connected, where every processor has n – 1 connections, one to each of the other processors, or other topology. A topology is characterized by its diameter, total bandwidth, and bisection bandwidth.

Systems may also be classified by their architectures. Uniform memory access (UMA), are systems that can be achieved only by a shared memory system, in which the memory is not physically distributed. A system that does not have this property is known as a nonuniform memory access (NUMA). Distributed memory systems have non-uniform memory access. Although simpler to design and build, non-cache-coherent NUMA systems become prohibitively complex to program in the standard von Neumann architecture programming model. As a result, all NUMA computers sold to the market use special-purpose hardware to maintain cache coherence, are classified as cache coherent NUMA, or CC-NUMA. In cache only memory access (COMA), local memories (typically DRAM) at each node are used as cache. This is in contrast to using the local memories as actual main memory, as in NUMA organizations. Network or cluster of workstations (NOW or COW, respectively), are simply a series of networked or clustered workstations linked together to increase their performance speed. Massively parallel processor (MPP), are computer systems with many independent arithmetic units or entire microprocessors that run in parallel. The term “massive” connotes hundreds if not thousands of such units. Some multiprocessors, including dataflow computers, systolic arrays, and neural networks, do not fit easily into any of these classifications.

Systems may use fixed communication connections, as in clustering, or reconfigurable connections. Reconfigurable communications systems are implemented using crossbar switches or multistage interconnection networks. Some MINs, such as the Benes network, are rearrangable but require a centralized routing unit. Others, such as the Omega network, are more limited but can be distributed, self-routing.

Multiprocessor systems often use shared memory. Memory interleaving is used to partition memory so that different modules can be accessed simultaneously. The primary purpose of interleaving is to adjust the timing differences between where the computer is ready to transfer data, and when that data is actually arriving at the drive head to be read. Each processor in a multiprocessor system may have its own cache memory. Protocols such as the MESI protocol (or Illinois protocol) are used to maintain coherence among data in the caches. In the MESI protocol, every cache line is marked with one of the four following states: Modified, Exclusive, Shared, or Invalid. The Modified state means the chance line is present only in the current cache, and is dirty; it has been modified from the value in main memory. The cache is required to write the data back to main memory at some time in the future, before permitting any other read of the main memory state. The Exclusive state is where the cache line is present only in the current cache, but is clean; it matches main memory. It may be changed to the Shared state at any time, in response to a read request. The Shared state indicates that this cache line may be stored in other caches of the machine and is clean; it matches the main memory. The line may be discarded at any time. Finally, the Invalid state indicates that this cache line is invalid.

In addition to the functions performed in uniprocessor systems, a multiprocessor operating system must avoid memory deadlocks among the processes being executed by its different processors. It also implements load balancing to maximize system performance.

Programming languages implement parallelism by incorporating explicit parallel constructs or by using libraries with parallel implementations. Conditional branch instructions and data dependencies inhibit parallelism in programs. Properly mapping a parallel algorithm to the multiprocessor’s topology and architecture maximizes system performance.