

The background features a light blue gradient. Three large, overlapping blue circles are arranged in a diagonal line from the top right to the bottom right. Two thin blue lines intersect at the top left and extend towards the circles. The text is positioned in the lower-left area.

Multiprocessing System

System Programming Seminar

Multiprocessing

Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them. There are many variations on this basic theme, and the definition of multiprocessing can vary with context, mostly as a function of how CPUs are defined.

Multiprocessing sometimes refers to the execution of multiple concurrent software processes in a system as opposed to a single process at any one instant. However, the terms multitasking or multiprogramming are more appropriate to describe this concept, which is implemented mostly in software, whereas multiprocessing is more appropriate to describe the use of multiple hardware CPUs. A system can be both multiprocessing and multiprogramming, only one of the two, or neither of the two.

Types

Processor symmetry

In a **multiprocessing** system, all CPUs may be equal, or some may be reserved for special purposes. A combination of hardware and operating-system software design considerations determine the symmetry (or lack thereof) in a given system. For example, hardware or software considerations may require that only one CPU respond to all hardware interrupts, whereas all other work in the system may be distributed equally among CPUs; or execution of kernel-mode code may be restricted to only one processor (either a specific processor, or only one processor at a time), whereas user-mode code may be executed in any combination of processors. Multiprocessing systems are often easier to design if such restrictions are imposed, but they tend to be less efficient than systems in which all CPUs are utilized.

Systems that treat all CPUs equally are called symmetric multiprocessing (SMP) systems. In systems where all CPUs are not equal, system resources may be divided in a number of ways, including asymmetric multiprocessing (ASMP), non-uniform memory access (NUMA) (multiprocessing), and clustered multiprocessing.

Processor coupling

Tightly-coupled multiprocessor systems contain multiple CPUs that are connected at the bus level. These CPUs may have access to a central shared memory (SMP or UMA), or may participate in a memory hierarchy with both local and shared memory (NUMA).

Chip multiprocessors, also known as multi-core computing, involves more than one processor placed on a single chip and can be thought of the most extreme form of tightly-coupled multiprocessing. Mainframe systems with multiple processors are often tightly-coupled.

Loosely-coupled multiprocessor systems (often referred to as clusters) are based on multiple standalone single or dual processor commodity computers interconnected via a high speed communication system (Gigabit Ethernet is common). (A Linux Beowulf cluster is an example of a loosely-coupled system).

Tightly-coupled systems perform better and are physically smaller than loosely-coupled systems, but have historically required greater initial investments and may depreciate rapidly; nodes in a loosely-coupled system are usually inexpensive commodity computers and can be recycled as independent machines upon retirement from the cluster.

Power consumption is also a consideration .Tightly-coupled systems tend to be much more energy efficient than clusters .This is because considerable economies can be realized by designing components to work together from the beginning in tightly-coupled systems, whereas loosely-coupled systems use components that were not necessarily intended specifically for use in such systems.

Instruction and data streams

In multiprocessing, the processors can be used to execute a single sequence of instructions in multiple contexts)single-instruction, multiple-data or SIMD, often used in vector processing(, multiple sequences of instructions in a single context)multiple-instruction, single-data or MISD(, or multiple sequences of instructions in multiple contexts)multiple-instruction, multiple-data or MIMD)

Software implementation issues

Flynn's taxonomy

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

SISD multiprocessing

In computing, **SISD** **S**ingle **I**nstruction, **S**ingle **D**ata (is a term referring to a computer architecture in which a single processor, a uniprocessor, executes a single instruction stream, to operate on data stored in a single memory .

In a single instruction stream, single data stream computer one processor sequentially processes instructions; each instruction processes one data item.

SISD is one of the four main classifications .In this system classifications are based upon the number of concurrent instructions and data streams present in the computer architecture .SISD can have concurrent processing characteristics .Instruction fetching and pipelined execution of instructions are common examples found in most modern SISD computers.

SIMD multiprocessing

In a single instruction stream, multiple data stream computer one processor handles a stream of instructions, each one of which can perform calculations in parallel on multiple data locations.

SIMD multiprocessing is well suited to parallel or vector processing, in which a very large set of data can be divided into parts that are individually subjected to identical but independent operations. A single instruction stream directs the operation of multiple processing units to perform the same manipulations simultaneously on potentially large amounts of data.

For certain types of computing applications, this type of architecture can produce enormous increases in performance, in terms of the elapsed time required to complete a given task. However, a drawback to this architecture is that a large part of the system falls idle when programs or system tasks are executed that cannot be divided into units that can be processed in parallel.

Additionally, programs must be carefully and specially written to take maximum advantage of the architecture, and often special optimizing compilers designed to produce code specifically for this environment must be used. Some compilers in this category provide special constructs or extensions to allow programmers to directly specify operations to be performed in parallel.

SIMD multiprocessing finds wide use in certain domains such as computer simulation, but is of little use in general-purpose desktop and business computing environments.

MISD multiprocessing

In computing, **MISD** (Multiple Instruction, Single Data) is a type of parallel computing architecture where many functional units perform different operations on the same data. Pipeline architectures belong to this type, though a purist might say that the data is different after processing by each stage in the pipeline.

MISD multiprocessing offers mainly the advantage of redundancy, since multiple processing units perform the same tasks on the same data, reducing the chances of incorrect results if one of the units fails. MISD architectures may involve comparisons between processing units to detect failures. Fault-tolerant computers executing the same instructions redundantly in order to detect and mask errors, in a manner known as task replication, may be considered to belong to this type.

Apart from the redundant and fail-safe character of this type of multiprocessing, it has few advantages, and it is very expensive. It does not improve performance. Not many instances of this architecture exist, as MIMD and SIMD are often more appropriate for common data parallel techniques. Specifically, they allow better scaling and use of computational resources than MISD does.

MIMD multiprocessing

In computing, **MIMD** (Multiple Instruction stream, Multiple Data stream) is a technique employed to achieve parallelism. Machines using MIMD have a number of processors that function asynchronously and independently. At any time, different processors may be executing different instructions on different pieces of data. MIMD

architectures may be used in a number of application areas such as computer-aided design/computer-aided manufacturing, simulation, modeling, and as communication switches .MIMD machines can be of either shared memory or distributed memory categories .These classifications are based on how MIMD processors access memory.

MIMD multiprocessing architecture is suitable for a wide variety of tasks in which completely independent and parallel execution of instructions touching different sets of data can be put to productive use .For this reason, and because it is easy to implement, MIMD predominates in multiprocessing.

MIMD does raise issues of deadlock and resource contention, however, since threads may collide in their access to resources in an unpredictable way that is difficult to manage efficiently .MIMD requires special coding in the operating system of a computer but does not require application changes .Both system and user software may need to use software constructs such as *semaphores*) also called *locks* or *gates* (to prevent one thread from interfering with another if they should happen to cross paths in referencing the same data .This gating or locking process increases code complexity, lowers performance, and greatly increases the amount of testing required, although not usually enough to negate the advantages of multiprocessing.

Symmetric multiprocessing

In computing, **symmetric multiprocessing** or **SMP** involves a multiprocessor computer architecture where two or more identical processors can connect to a single shared main memory .Most common multiprocessor systems today use an SMP architecture .In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors.

SMP systems allow any processor to work on any task no matter where the data for that task are located in memory; with proper operating system support, SMP systems can easily move tasks between processors to balance the workload efficiently.

SMP represents one of the earliest styles of multiprocessor machine architectures, typically used for building smaller computers with up to 8 processors .Larger computer systems might use newer architectures such as NUMA (Non-Uniform Memory Access)

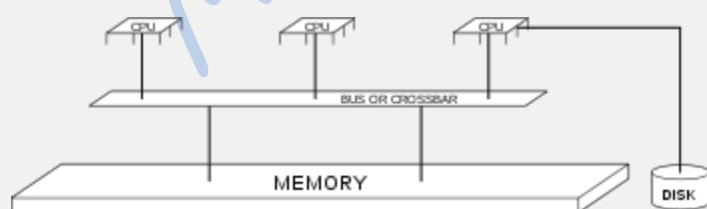


Diagram of a typical SMP system .Three processors are connected to the same memory module through a bus or crossbar switch

Asymmetric multiprocessing

Asymmetric multiprocessing varies greatly from the standard processing model that we see in personal computers today .Due to the complexity and unique nature of this architecture, it was not adopted by many vendors or programmers during its brief stint (1970 – 1980)

Whereas a symmetric multiprocessor or SMP treats all of the processing elements in the system identically, an ASMP system assigns certain tasks only to certain processors.

Asymmetric hardware systems commonly dedicated individual processors to specific tasks .For example, one processor may be dedicated to disk operations, another to video operations, and the rest to standard processor tasks .These systems don't have the flexibility to assign processes to the least-loaded CPU, unlike an SMP system.

Although hardware-level ASMP may not be in use, the idea and logical process is still commonly used in applications that are multiprocessor intensive .Unlike SMP applications, which run their threads on multiple processors, ASMP applications will run on one processor but outsource smaller tasks to another .Although the system may physically be an SMP, the software is still able to use it as an ASMP by simply giving certain tasks to one processor and deeming it the "master", and only outsourcing smaller tasks to "slave "processors.

Hardware ASMP

Overview

Asymmetrical multiprocessors are defined by the characteristic that each processor is unique (non-symmetrical) . It is common to have one processor that has access to the memory map as a whole, and other processors which simply act as slaves to the main or master processor .Usually, these slave processors will have their own memory which is not tied to the primary processors memory .Slave processors are required to exchange data with the main processor through a partitioned segment of memory that is allocated solely for the purpose of communication . Depending on the hardware in question, each processor may or may not be able to speak to other processors directly.

Differences between hardware ASMP and SMP

In the symmetrical multiprocessing design, each processor is able to access the entire memory map; there are no master or slave processors .In this case each processor is non-unique and has equal power .This means that they can share memory between themselves and can interact with each other directly, regardless of how many there are in the system .People commonly confuse these architectures and as such it is important to define the differences.

Software ASMP

Overview

Asymmetric multiprocessing)as opposed to asymmetrical multiprocessors (is the term that refers to software side ASMP .In software each program or application is a process .ASMP for software means that all tasks/processes are unique .Thus a given task)such as your operating system or favourite game (would be assigned to a certain processor .In a more general context "a certain task not runs on every processor ."It is common for application

which uses ASMP to work in the following way .The main processor will determine what work needs to be done and will take the bulk of the load, from there it can create instances of the given task on other processors to complete work .Take a video rendering program, the main processor could run the application and the user interface, while offloading the rendering component to a slave processor .This type of action needs to be written into the software and is not decided at the hardware level .It is the programmers' responsibility to determine what jobs should be completed by a given processor.

It must be noted that most applications will ONLY run on the master processor and that the slave processors can merely take on the role of completing tasks that the master processor asks .It is rare that an entire application will or can be run from a slave processor.

Differences between software ASMP and SMP

Symmetrical multiprocessing, when referring to software, implies the exact opposite of ASMP. In regards to the operating system, a SMP machine is able to spawn any process/task on any of the processors available .Because SMP systems have no master or slave processors, each logical unit is able to complete a given task .In an ASMP system, a certain processor may not be able to complete a task for a number of reason such as the inability to access the entire memory map, special purpose nature of the processor)e.g .a coprocessor(, and thus tasks must be give to it by master processor .Therefore, it is up to the programmer to make sure the processors are being used to their maximum potential .In an ASMP environment, a programmer has to worry about whether a processor can complete a given task and how to make the processors communicate effectively to distribute tasks.

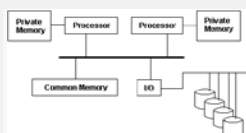
Modern applications of ASMP

Currently there are no consumer level production computers that use asymmetric multiprocessor designs .There are, however, computers that are able to distribute tasks Asymmetrically .In theory you are able to use a Symmetrical processor to do asymmetrical computations .A programmer can choose to use one processor as a main, and only offload certain tasks to the other processor .Although each physical or logical processor is able to complete any given task, priority is given to one as the "master "processor, and the other is given the position of "slave."

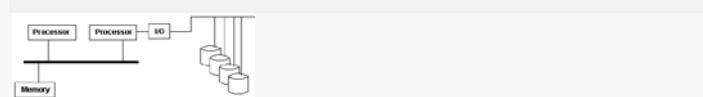
Graphical representation of asymmetric multiprocessing

Below are examples of what a cluster of asymmetrical multiprocessors would look like .Observe the extremely unique nature of these designs and how only one processor has access to the I/O part of the system .As stated before, these systems work best and were originally designed to do very specific tasks .One processor may simply do physics calculations while another is dedicated to rendering 2D video .Above those two processors, will be a master processor that assigns tasks.

Notice also that the main memory is not accessible by all of the processors .The master processor will usually relay information on a "need to know "basis, to the slave processors.



Multiple processors with unique access to memory and I/O.



This image depicts an ASMP system where only one processor has direct access to I/O

Multiprocessing System

Prepared By:

- Noora Sadon.
- Rand Nawfal.
- Karar Shakir.
- Muhanned Raad.