Abstract

This paper includes the current researches in context to parallel computer architecture or the parallel architecture of computers. The different architectures viz. SISD, SIMD, MISD, MIMD are explained and compared to each other , for having an idea that which one is suited for what kind of different application areas requiring the multiprocessor architectures. Multiprocessing architectures provide hardware for executing multiple tasks simultaneously via techniques such as simultaneous multithreading and symmetric multiprocessing. Synchronization is a crucial operation in many parallel applications. Conventional synchronization mechanisms are failing to keep up with the increasing demand for efficient synchronization operations as systems grow larger and network latency increases.

1. Introduction

An organizational technique in which a number of processor units are employed in a single computer system to increase the performance of the system in its application environment above the performance of a single processor of the same kind. In order to cooperate on a single application or class of applications, the processors share a common resource. Usually this resource is primary memory, and the multiprocessor is called a primary memory multiprocessor. A system in which each processor has a private (local) main memory and shares secondary (global) memory with the others is a secondary memory multiprocessor, sometimes called a multicomputer system because of the looser coupling between processors.

Mode of computer operation in which two or more processors (CPU) are connected and are active at the same time. In such a system, each processor is executing a different program or set of instructions, thus increasing computation speed over a system that has only one processor (which means only one program can be executed at a time). Because the processors must sometimes access the same resource (as when two processors must write to the same disk), a system program called the task manager has to coordinate the processors' activities.

Multiprocessor systems may be classified into types: single instruction stream, single data stream (SISD); single instruction stream, multiple data stream (SIMD); multiple instruction stream, single data stream (MISD): and multiple instruction stream, multiple data stream (MIMD). Systems in the MISD category are rarely built. The other three architectures may be distinguished simply by the differences in their respective instruction cycles:

In an **SISD** architecture there is a single instruction cycle; operands are fetched in serial fashion into a single processing unit before execution. Sequential processors fall into this category.

An **SIMD** architecture also has a single instruction cycle, but multiple sets of operands may be fetched to multiple processing units and may be operated upon simultaneously within a single instruction cycle. Multiple-functional-unit, array, vector, and pipeline processors are in this category. See also Supercomputer.

In an **MIMD** architecture, several instruction cycles may be active at any given time, each independently fetching instructions and operands into multiple processing units and operating on them in a concurrent fashion. This category includes multiple processor systems in which each processor has its own program control, rather than sharing a single control unit.

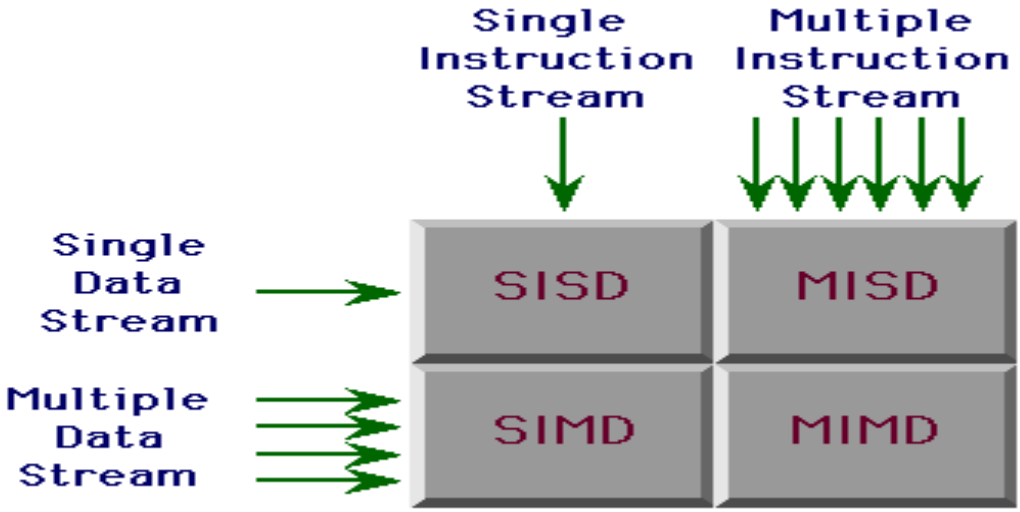
MIMD systems can be further classified into throughput-oriented systems, high-availability systems, and response-oriented systems. The goal of throughput-oriented multiprocessing is to obtain high throughput at minimal computing cost in a general-purpose computing environment by maximizing the number of independent computing jobs done in parallel. High-availability multiprocessing systems are generally interactive, often with never-fail real-time online performance requirements.

The goal of response-oriented multiprocessing (or parallel processing) is to minimize system response time for computational demands. Simultaneous processing with two or more processors in one computer or two or more computers processing

together. When two or more computers are used, they are tied together with a high-speed channel and share the general workload between them. If one fails, the other takes over. Multiprocessing is also accomplished in special-purpose computers, such as vector processors, which provide concurrent processing on sets of data.

1.1.Types of Multiprocessor Systems:

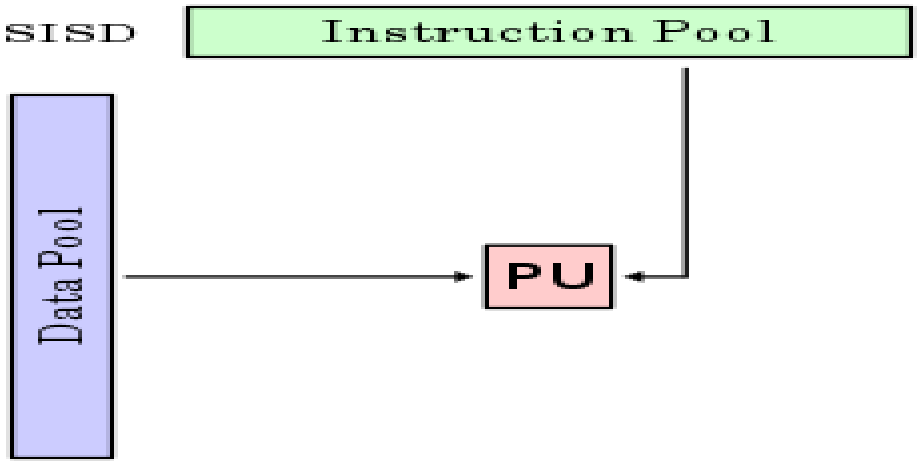
Multiprocessor systems may be classified into types: single instruction stream, single data stream (SISD); single instruction stream, multiple data stream (SIMD); multiple instruction stream, single data stream (MISD): and multiple instruction stream, multiple data stream (MIMD). Systems in the MISD category are rarely built. The other three architectures may be distinguished simply by the differences in their respective instruction cycles:



1.1.1.SISD (Single Instruction Single Data)

In a single instruction stream, single data stream computer one processor sequentially processes instructions, each instruction processes one data item. One example is the "von Neumann" architecture with RISC.

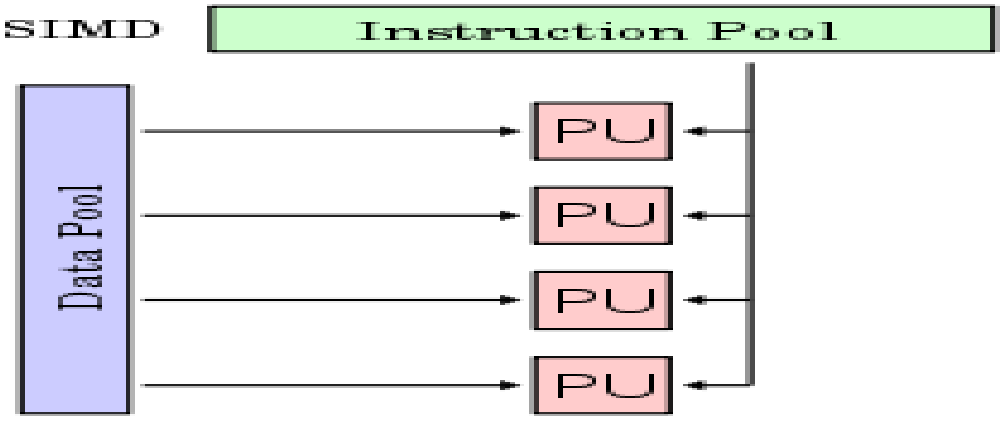
SIMD multiprocessing



1.1.2. SIMD (Single Instruction Multiple Data)

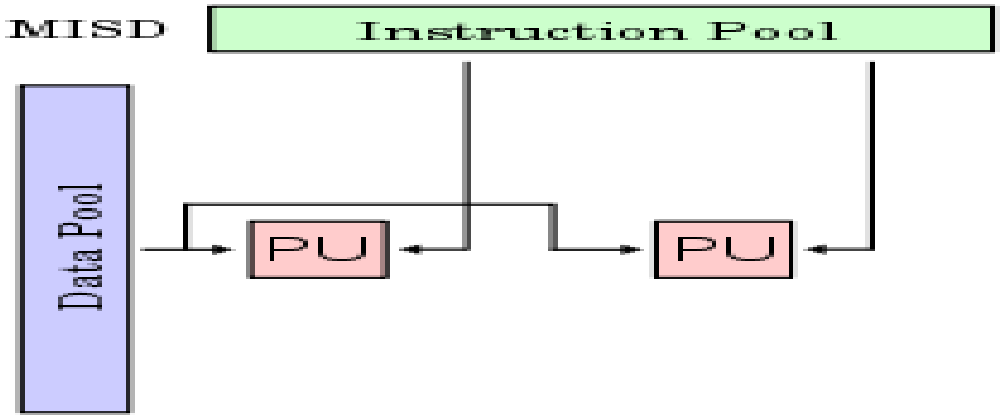
In a single instruction stream, multiple data stream computer one processor handles a stream of instructions, each one of which can perform calculations in parallel on multiple data locations.
SIMD multiprocessing is theyll suited to parallel or vector processing, in which a very large set of data can 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 (e.g., DO FOR ALL statements in the version of FORTRAN used on the ILLIAC IV, which was a SIMD multiprocessing supercomputer).

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.



1.1.3.**MISD (Multiple Instruction Single Data)**

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. 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. It can be implemented in a way that is transparent to software. It is used in array processors and is implemented in fault tolerant machines.
Another example of MISD is pipelined image processing where every image pixel is piped through several hardware units performing several steps of image transformation.
MIMD multiprocessing



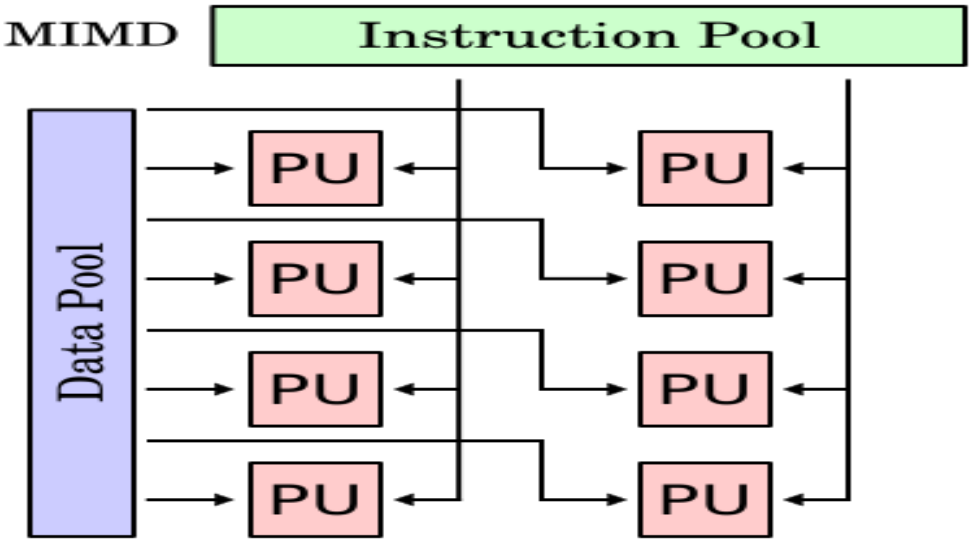
1.1.4.MIMD (Multiple Instruction Multiple Data)

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.

Processing is divided into multiple threads, each with its own hardware processor state, within a single software-defined process or within multiple processes. Insofar as a system has multiple threads awaiting dispatch (either system or user threads), this architecture makes good use of hardware resources.

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 unless the programs themselves use multiple threads (MIMD is transparent to single-threaded programs under most operating systems, if the programs do not voluntarily relinquish control to the OS). 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.

Similar conflicts can arise at the hardware level between processors (cache contention and corruption, for example), and must usually be resolved in hardware, or with a combination of software and



hardware

Goal

The goal of response-oriented multiprocessing (or parallel processing) is to minimize system response time for computational demands.[3]

2.Literature Survey

2.1.RECENT PROGRESS IN MULTIPROCESSOR THREAD SCHEDULING
Daniel Shapiro

In this paper they discuss progress in the area of thread scheduling for multiprocessors, including systems which are Chip-MultiProcessors (CMP), can perform Simultaneous MultiThreading (SMT), and/or support multiple threads to execute in parallel. The reviewed papers approach thread scheduling from the aspects of resource utilization, thread priority, Operating System (OS) effects, and interrupts. The metrics used by the discussed papers will be summarized.
Index Terms— Scheduling, multiprocessor, threads, priority.

2.2. Using Symmetric Multiprocessor Architectures formHigh Performance Computing Environments
Mohsan Tanveer, M. Aqeel Iqbal, Farooque Azam

Performance enhancement for high speed computing can be carried out by using many techniques and architectures at software and high hardware level. Performance enhancement using hardware techniques may include the use of multiple computing nodes or a single node consisting of multiple processors. Symmetric multiprocessor is one of the modern architectures used to perform extensive computations.Symmetric multiprocessors have many configuration modes to carry out these heavy computations. The performance of Symmetric multiprocessors is analyzed and compared with high-fidelity models. Processors models are used to design and construct the architectures of symmetric multiprocessors. In this research paper such kind of critical design aspects of symmetric multi processors have been analyzed.

2.3. INTRODUCTION TO MULTIPROCESSOR I/O ARCHITECTURE
David Kotz
The computational performance of multiprocessors continues to improve by leaps and bounds fueled in part by rapid improvements in processor and interconnection technology I/O performance thus becomes ever more critical to avoid becoming the bottleneck of system performance In this paper they provide an introduction to I/O architectural issues in multiprocessors with a focus on disk subsystems While they discuss examples from actual architectures and provide pointers to interesting research in the literature they do not attempt to provide a comprehensive survey. They concentrate on a study of the architectural design issues and the effects of different design alternatives

2.4. Fast Synchronization on Shared-Memory Multiprocessors: An Architectural Approach
Zhen Fang1, Lixin Zhang2, John B. Carter1, Liqun Cheng1, Michael Parker3

Synchronization is a crucial operation in many parallel applications. Conventional synchronization mechanisms
are failing to keep up with the increasing demand for efficient synchronization operations as systems grow larger and network latency increases. The contributions of this paper are threefold. First, they revisit some representative synchronization algorithms in light of recent architecture innovations and provide an example of how the simplifying assumptions made by typical analytical models of synchronization mechanisms can lead to significant performance estimate errors. Second, they present an architectural innovation called active memory that enables very fast atomic operations in a shared-memory multiprocessor. Third, they use execution-driven simulation to quantitatively compare the performance of a variety of synchronization mechanisms based on both existing hardware techniques

and active memory operations. To the best of their knowledge, synchronization based on active memory outforms all existing spinlock and non-hardwired barrier implementations by a large margin.
Keywords: distributed shared-memory, coherence protocol, synchronization, barrier, spinlock, memory
controller

2.5. Temporal Isolation on Multiprocessing Architectures
Dai Bui, Edward A. Lee , Isaac Liu , Hiren D. Patel , Jan Reineke

Multiprocessing architectures provide hardware for executing multiple tasks simultaneously via techniques such as simultaneous multithreading and symmetric multiprocessing. The problem addressed by this paper is that even when tasks that are executing concurrently do not communicate, they may interfere by affecting each other's timing. For cyber-physical system applications, such interference can nullify many of the advantages o ered by parallel hardware. In this paper, they argue for temporal semantics in layers of abstraction in computing. This will enable us to achieve temporal isolation on multiprocessing architectures. They discuss techniques at the micro-architecture level, in the memory hierarchy, in on-chip communication, and in the instruction-set architecture that can provide temporal semantics and control over timing.
Keywords:
Precision-timed architectures, micro-architecture, pipelines, memory hierarchy, network on chip, instruction set architecture

3. Methodology Used
Concurrent Instruction Processing
All computers perform simultaneous functions, such as executing instructions while reading from an input device and writing to an output device. CPUs can also execute multiple instructions simultaneously from a single stream of instructions (see pipeline processing). However, multiprocessing refers specifically to the concurrent execution of two or more independent streams of instructions. See parallel processing, SMP, MPP, CMP, bus mastering and fault tolerant.
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 (multiple cores on one die, multiple dies in one package, multiple packages in one system unit, etc.).
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 of them.

3.1Types
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.

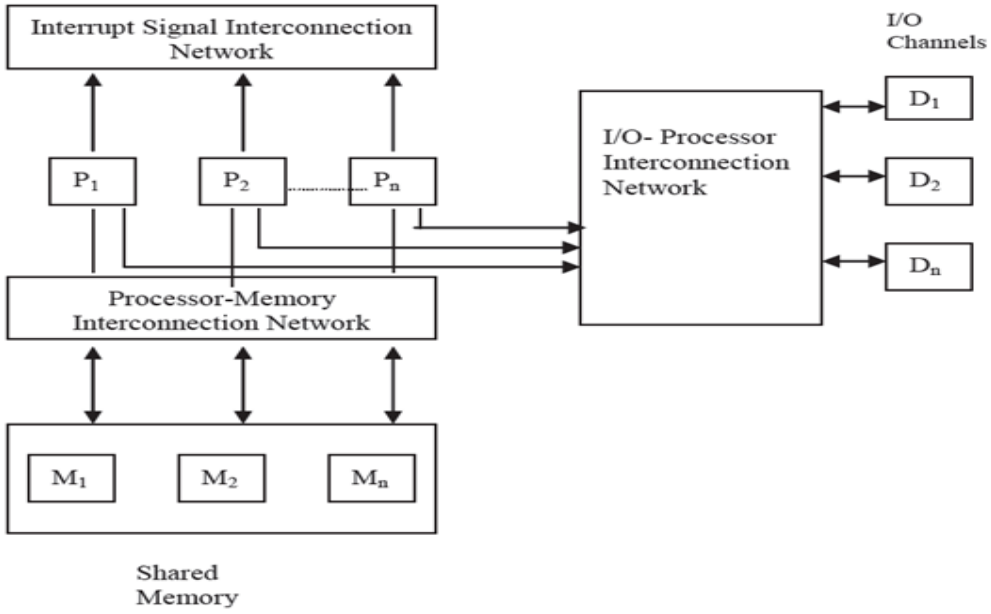
Instruction and data streams

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, used for redundancy in fail-safe systems and sometimes applied to describe pipelined processors or hyper-threading), or multiple sequences of instructions in multiple contexts (multiple-instruction, multiple-data or MIMD).

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). The IBM p690 Regatta is an example of a high end SMP system. Intel Xeon processors dominated the multiprocessor market for business PCs and were the only x86 option until the release of AMD's Opteron range of processors in 2004. Both ranges of processors had their own onboard cache but provided access to shared memory; the Xeon processors via a common pipe and the Opteron processors via independent pathways to the system RAM.

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.

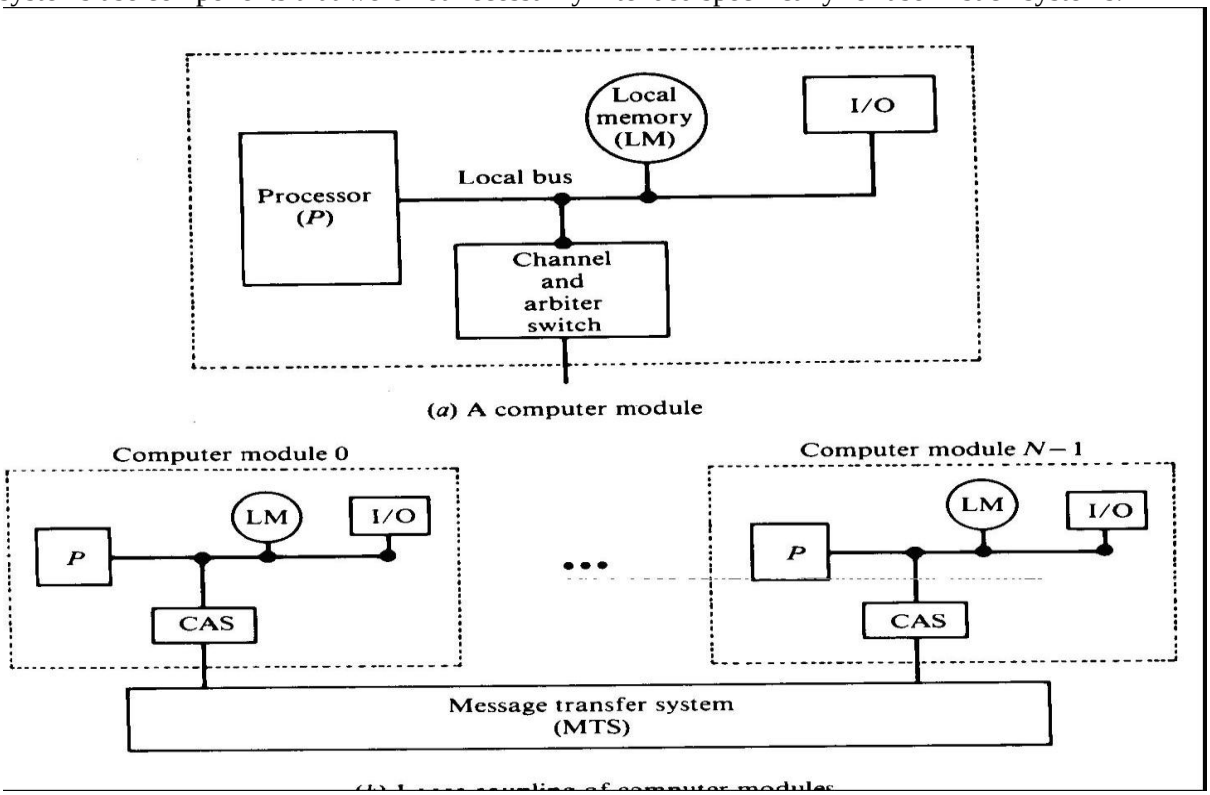


Loosely-coupled multiprocessor systems (often referred to as clusters) are based on multiple standalone single or dual processor commodity computers interconnected via a high speed communication system (Gigabit Ethernet is common). A Linux Beowulf cluster is an example of a loosely-coupled system.

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 economy 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.



4. Gaps in Study

An attempt has been made to classify the different multiprocessor architecture systems, with the help of required diagrams and comparisons but there is a limitation to scalable, shared memory architectures and problem of cache coherence or memory coherence.

5. Summary

This paper has attempted to describe the four major classifications of the multiprocessor system architectures. Required comparisons have been made. The classification and differentiation between these types has been made with the help of required diagrams, for having an idea that which one is suited for what kind of different application areas requiring the multiprocessor architectures.

6. Future Scope

This paper has attempted to describe the four major classifications of the multiprocessor system architectures. Required comparisons have been made. The classification and differentiation between these types has been made with the help of required diagrams, for having an idea that which one is suited for what kind of different application areas requiring the multiprocessor architectures. An attempt will be made to include the leftover topics like scalable, shared memory architectures and problem of cache coherence or memory coherence, to make the research complete.

References

1.RECENT PROGRESS IN MULTIPROCESSOR THREAD SCHEDULING
Daniel Shapiro

2. Using Symmetric Multiprocessor Architectures for
High Performance Computing Environments
Mohsan Tanveer, M. Aqeel Iqbal, Farooque Azam

3. INTRODUCTION TO MULTIPROCESSOR I/O ARCHITECTURE
David Kotz
4. Fast Synchronization on Shared-Memory Multiprocessors: An Architectural Approach
Zhen Fang1, Lixin Zhang2, John B. Carter1, Liqun Cheng1, Michael Parker3

5. Temporal Isolation on Multiprocessing Architectures
Dai Bui, Edward A. Lee , Isaac Liu , Hiren D. Patel , Jan Reineke