Eno Asuquo *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 8, August- 2022, pg. 63-71 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Parallel Computational Models

Eno Asuquo[1]; V.I.E. Anireh (Ph.D)[2]
[1] enoasuquo8@gmail.com
[2] anireh.ike@ust.edu.ng

DOI: 10.5281/zenodo.7025236

Abstract
The purpose of this study is to examine the advantages of using parallel computing. The phrase "parallel computing" refers to a strategy for allocating all of a system's resources in order to maximize performance and programmability while abiding by time and cost constraints. The main driving forces are to improve performance, cut costs, and deliver precise outcomes. Look-ahead, Pipeline, Vectorization, Concurrency, Multitasking, Multiprogramming, Time Sharing, Multi-Threading, and Distributed systems are just a few of the methods that can be used to show parallelism. By dividing a task into its component pieces and allocating each of those parts to a different processor, parallel computing can be accomplished in order to reduce the amount of time required to finish a program.

Keywords: Parallelism, Computational Models, Multiprocessor, Multicomputers

1. Introduction
Parallel processing is a useful method that can be found on modern computers. It was developed as a result of efforts to achieve faster speeds at lower costs and more accurate results in applications that are used in the real world. This quest led to the development of an effective technique that can be found in modern computers. As a result of the practice of multiprogramming, which can also be referred to as multiprocessing or multicomputing, modern computers are capable of handling multiple tasks at the same time, making concurrent events extremely prevalent.
On modern personal computers, you can find installable software packages that are both powerful and extensive. In order to explore the rise of computer performance, we first need to have a fundamental grasp of the advancements achieved in both computer hardware and software. Only then can we begin to investigate the growth of computer performance.
The concept of multidimensionality can be illustrated using a wide variety of methods, including look-ahead processing, pipelining, vectorization, concurrency, simultaneity, data parallelism, partitioning, interleaving, overlapping, multiplicity, replication, time sharing, space sharing, multitasking, multi-programming, multi-threading, and distributed computing, to name a few.

2. Computer Development Milestones
The mechanical and electrical stages of computer development were, respectively, the first two significant stages of the field's evolution. Before 1945, computers were either of a mechanical or electromechanical design, depending on the historical period in which they were developed. The abacus, which was invented in China about the year 500 B.C., is recognized as the world's first example of a mechanical computer. Calculating decimal arithmetic by hand using an abacus, including the propagation of carry values from one digit to the next. In the year 1642, the French city of Prance was the location of Blaise Pascal's invention of a mechanical

adder and subtractor. Charles Babbage, working in England in the year 1827, devised a difference engine with the intention of using it for polynomial evaluation. In the year 1941, German inventor Konrad Zuse created the very first binary mechanical computer. This accomplishment took place in Germany. Howard Aiken was the brains behind the creation of the first electromechanical decimal computer, which was produced by IBM in 1944 and given the name Harvard Mark I. The computers designed by Zuse and Aiken were both created with the intention of carrying out computations that might be used in a wide range of different domains. It should not come as a complete surprise that the fact that processing and communication were achieved with moving mechanical parts significantly slowed down the calculation speed of mechanical computers and made them significantly less reliable. The first time that electrical circuitry was used can be considered the starting point for the development of contemporary computers. Electrons, which possess a high degree of mobility, have taken the place of the moving parts that were formerly found in mechanical computers in electronic computers. Transmission of information has mostly transitioned away from the use of mechanical gears and levers in favor of the transmission of information by electric signals, which can move at nearly the speed of light.

3. Computer Generations

Over the course of the previous half century, there have been five major generations of advances made in the field of electronic computers. The progression of electronic computers is split down into five separate generations in Table 1.

Table1: Five Generations of Electronic Computer Development

Generation	**Technology and Architecture**	**Software and Application**	**Representative System**
First (1945-54)	Vacuum tubes and relav memories, CPU driven by PC and accumulator, fixed-point arithmetic.	Machine/assembly languages, single user, no subroutine linkage, programmed I/O using CPU.	ENIAC, Princeton IAS, IBM 701.
Second (1955-64)	Discrete transistors and core memories, floating-point arithmetic, I/O processors, multiplexed memory access.	HLL used with compilers, subroutine libraries, batch processing monitor.	IBM 7090, CDC 1604, Univac LARC.
Third (1965-74)	Integrated circuits (SSI/-MSI), microprogramming, pipelining, cache, and lookahead processors.	Multiprogramming and timesharing OS, multiuser applications.	IBM 360/370, CDC 6600, TI-ASC, PDP-8.
Fourth (1975-90)	LSI/VLSI and semiconductor memory, multiprocessors, vector supercomputers, multicomputers	Multiprocessor OS, languages, compilers, and environments for parallel processing	VAX 9000, Cray X-MP, IBM 3090, BBN TC2000.
Fifth (1991- present)	ULSI/VHSIC processors, memory, and switches, high-density packaging, scalable architectures.	Massively parallel processing, grand challenge applications, heterogeneous processing.	Fujitsu VPP500, Cray/MPP, TMC/CM-5, Intel Paragon

4. Elements of Modern Computers

Over the course of the previous half century, there have been five major generations of advances made in the field of electronic computers. The progression of electronic computers is split down into five separate generations in Table 1.A user interface, computer hardware, instruction sets, application programs, and

operating system software are the components that come together to form a contemporary computer system. A modern computer system will also typically have something called a user interface.

In this part, the components of a modern computer system, including their hardware, software, and programming, are dissected and examined in a nutshell in relation to parallel processing. These components include: Since quite some time ago, it has been generally understood that the idea of computer architecture is no longer limited to the structure of bare machine hardware. This is something that has been universally acknowledged. A modern computer is a self-contained system that is comprised of the following components: machine hardware, an instruction set, system software, application programs, and user interfaces. Each of these components is essential to the operation of the computer. Figure 1 illustrates several parts of the system that make up the whole. When dealing with real-world scenarios, the utilization of a computer is obligatory.

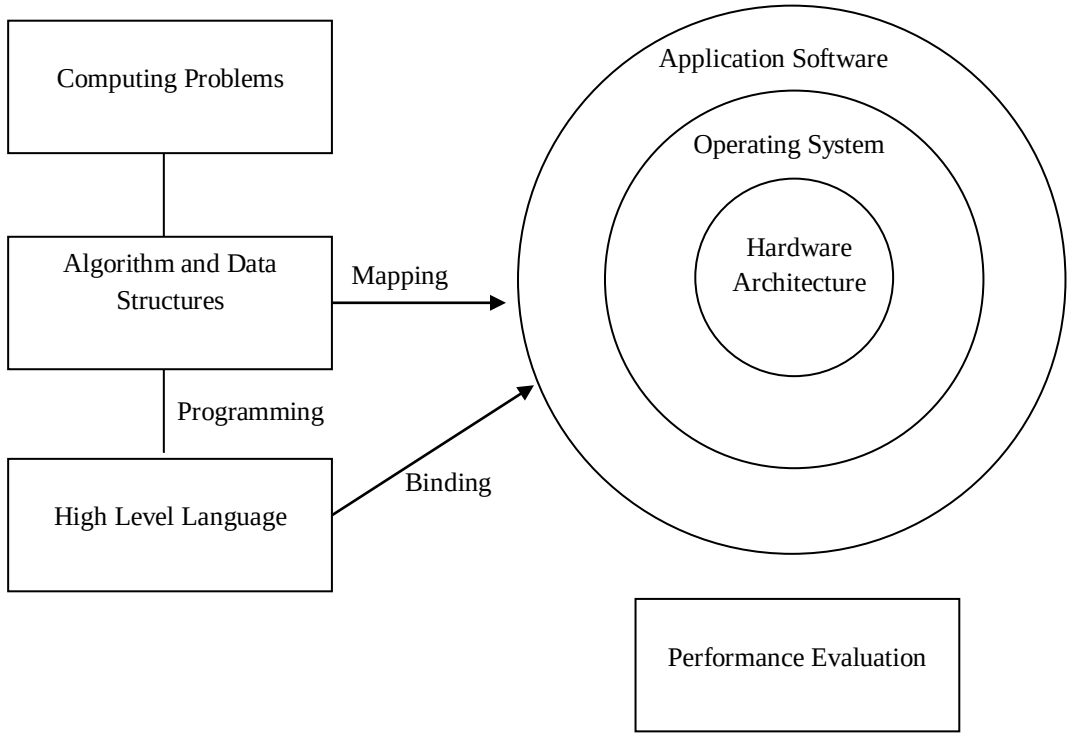

Figure 1: Elements of a Modern Computer System

Problems that can be solved using computation, problems that require logical thinking, and problems that involve transaction processing are the three primary types of computer problems. It is possible that in order to successfully execute certain demanding activities, the usage of all three modes of processing will be required.

- The Evolving Nature of Computer Architecture over the Course of Time: Significant shifts have come about as a result of the breakthroughs that have been made in computer architecture over the course of the past four decades. As the complexity of our computing needs increased, we transitioned away from the Von Neumann architecture and toward the use of several computers and processors.

- The functionality of a computer or computer system depends on a computer system's performance is determined not only by the capabilities of the machine itself but also by the operations of the programs that are currently being executed on the system. Utilizing higher hardware technology, incorporating more complex design features, and effectively managing available resources are all ways in which the capabilities of a computer can be enhanced.

- Since the behavior of the software is dependent on application variables as well as run-time variables, it is impossible to predict how it will behave.

5. Multiprocessors and Multi-computers

This section will go through two types of parallel computers: Shared-Memory Multiprocessors and Message Passing Multicomputers

5.1 Shared-Memory Multi-computers

The following are the three multiprocessor models with shared memory that are used most frequently:

5.1.1 Uniform Memory Access (UMA)

The shared physical memory in this model is accessible to all of the model's processors, and each processor has access to the same total amount of memory. Every single CPU has the exact same level of access to each and every one of the memory words. Every CPU has the potential to have its very own memory cache if they so want. The numerous peripheral devices fall under the purview of the same criterion in this regard. If all of the processors in a multiprocessor system have equal access to the various peripheral devices that are connected to the system, then we call that system symmetrical. When just a subset of the processors in a multiprocessor system have access to the system's peripheral devices, the system is said to have an asymmetric architecture.

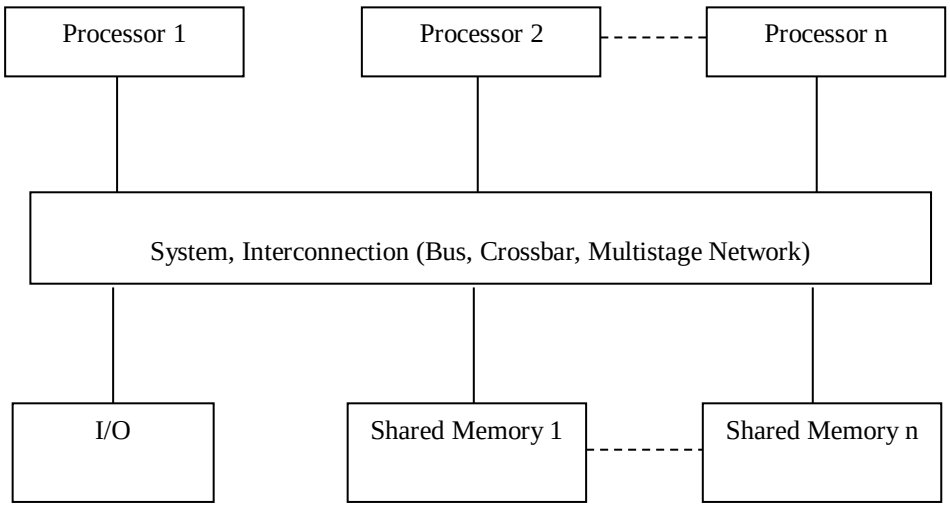

Figure 2: Uniform Memory Access

Eno Asuquo *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 8, August- 2022, pg. 63-71 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

5.1.2 Non-uniform Memory Access (NUMA)

Within the framework of the NUMA multiprocessor idea, the amount of time necessary to access a memory word is determined by the specific location of the word inside the memory. Local memory refers to the memory that is physically distributed throughout the system and is used by all of the processors. This memory is shared by all of the processors. When all of the local memory is combined, a global address space is generated, which all processors are able to make use of. This area can be used to store data.

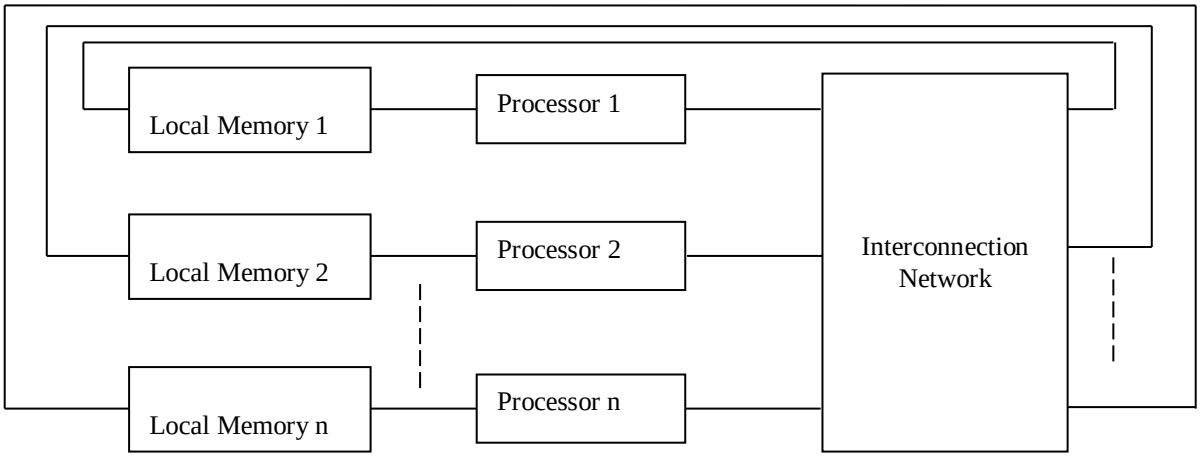

Figure 3: Non-uniform Memory Access

5.1.3 Cache Only Memory Architecture (COMA)

The NUMA model served as the foundation for the COMA paradigm, which was developed through further research and development. The full contents of the distributed main memory are copied into the cache memory at this position.

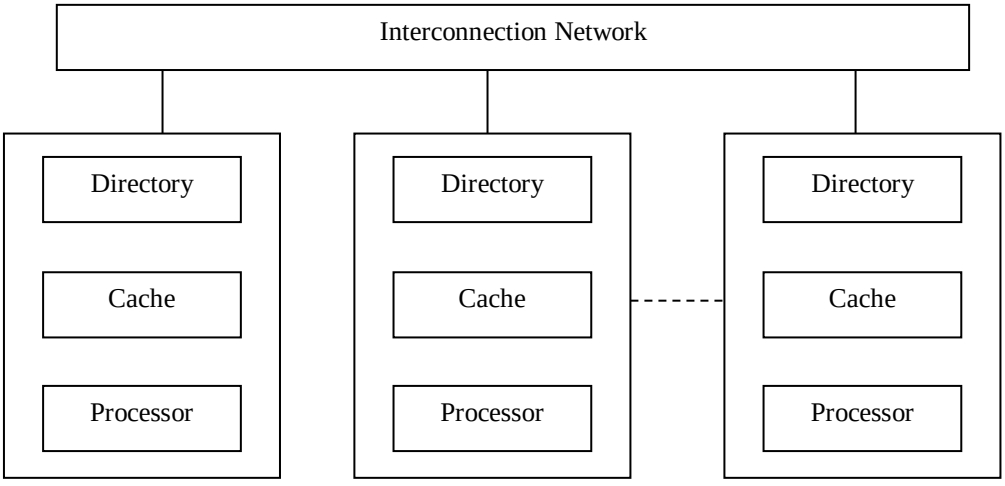

Figure 4: Cache Only Memory Architecture (COMA)

Eno Asuquo *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 8, August- 2022, pg. 63-71 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

5.2 Distributed - Memory Multi-computers

A distributed memory multi-computer system is comprised of a large number of individual computers, which are collectively referred to as nodes. These nodes are linked to one another by a network that is responsible for the transmission of messages. Each node acts as if it were its own standalone computer and is furnished with a CPU, local memory, and, in certain instances, input/output devices to carry out these functions. In this particular situation, all of the local memories are regarded as private, and the only processors that can access them are the ones that are located locally. Because of this, traditional computers are now more commonly referred to as no-remote-memory-access (NORMA) machines.

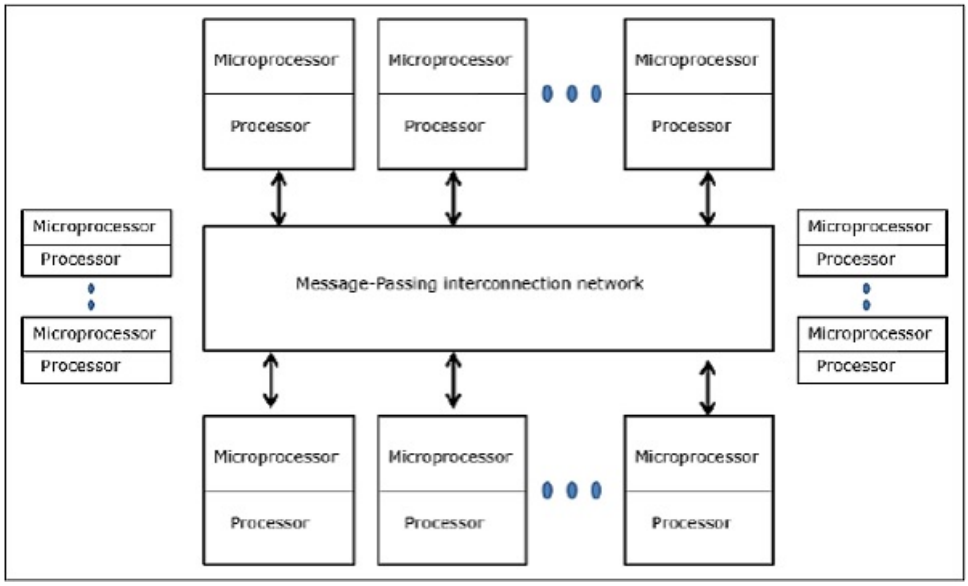

Figure 5: Distributed Multi-computer Architecture

5.2.1 Message Passing System (also referred to as distributed memory)

The functions of the local memory and the processor that can be found at each node of an interconnection network are typically combined as a matter of standard practice. Because there is no such thing as global memory, message forwarding is the only method that can be used to move data from one local memory to another. This is because there is no global memory. This is typically accomplished through the utilization of a Send/Receive pair of commands, both of which are required to be programmed into the application software in order for it to operate effectively. Because of this, programmers are required to have an understanding of the message-passing paradigm, which requires them to copy data and address issues of consistency. This is a consequence of the fact that this paradigm exists. Commercial examples of message forwarding architectures around the year 1990 included the nCUBE, the iPSC/2, and a huge variety of other computer-based systems. These examples were all available for purchase. These systems were eventually replaced by Internet-connected ones, the processor/memory nodes of which were either Internet servers or desktop clients. Eventually, these systems become obsolete. It was also abundantly clear that the utilization of distributed memory was the one and only option available for increasing the number of processors that could be effectively managed by a parallel and distributed system. This was the case due to the fact that it was the only option available. Computer systems absolutely need to implement distributed memory strategies in order to keep their ability to scale to ever

more complicated configurations. This is the only way for them to do so (as measured by the number of processors). The creation of systems that used the message passing paradigm allowed for greater scalability, but programming in the shared memory paradigm was easier to use. This resulted in a conflict between the two opposing forces. It was in computer systems such as the SGI Origin2000 that the distributed shared memory architecture, often known as DSM, was originally implemented for the first time. Memory is physically dispersed across these types of systems; for example, the hardware architecture adheres to the message passing school of thought, whereas the programming paradigm belongs to the shared memory school of thought. In its most fundamental form, software serves to conceal the hardware that it runs on. The design seems to a programmer to be a shared memory machine and operates in the same manner. However, in reality, it is a message-passing architecture that lies beneath the software. This gives the impression that the architecture is a shared memory machine. As a direct result of this, the DSM machine is a hybrid that takes cues from both schools of design in its creation. A message-passing system is a specific kind of multiprocessor that some computers have. Each processor on this sort of multiprocessor has access to its own local memory thanks to the multiprocessor's design. In contrast to shared memory systems, message-passing systems accomplish their communication by utilizing send and receive operations. This is in contrast to shared memory systems, which rely on a central memory location. In a network structured like this one, a node would consist of a central processing unit (CPU) and the memory that is associated with it. It is common for nodes to have the capacity to store messages in buffers, which are sectors of temporary memory where messages wait to be delivered or received, and to have the capacity to carry out send-and-receive operations concurrently with processing. Buffers are sectors of temporary memory that are used by nodes. Both the processing of messages and the computation of problems can be delegated to the underlying operating system, which is responsible for handling both tasks simultaneously. Every processor in the system has its own address space, and the processors do not share any global memory with one another. It is possible to connect the processing units of a message-passing system to one another using a variety of methods, ranging from architecture-specific interconnection structures to networks that are geographically distributed across the world. In principle, the method of delivering messages can scale up to support an enormously high number of users. Scalability is the capability of an application to grow the number of processors it employs without producing a visible decline in the overall operating efficiency of the application as a whole. In order for message-passing multiprocessors to connect with one another on a local level, they make use of a broad variety of static networks of varying configurations. It is impossible to exaggerate the significance of hypercube networks, particularly given the amount of attention that has been paid to them in recent years. Message-passing systems have also made use of close-neighbor networks that can operate in both two and three dimensions simultaneously. There are two key design factors that need to be given attention when creating interconnection networks for use in message transmission systems. One of them is security, and the other is performance. There is discussion of both network latency and link bandwidth in this passage. The greatest number of data packets that may be transmitted in a specific amount of time is referred to as the connection bandwidth, and it is expressed as bits per second (bits/s). Network latency refers to the amount of time it takes for a communication to be delivered to its intended recipient after it has been sent. Wormhole routing was first implemented in 1987 as an alternative to the usual store-and-forward routing that was used in the process of message transmission. The intention behind this change was to reduce the amount of message latency as well as the size of the buffer. During the process of wormhole routing, a packet is broken down into its individual pieces, which are referred to as flits (flow control bits). These flits go through the network in the form of a pipeline, with the header flit of the packet acting as the navigator to the node at which they are ultimately intended to arrive. Because of the congestion on the network, any flit that follows the header flit will also be stopped when it occurs. This is because the header flit comes first.

Eno Asuquo *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 8, August- 2022, pg. 63-71 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

6. Flynn's Taxonomy of Computer Architecture

Flynn created the taxonomy of computer architecture that is today the one that is most widely used in the year 1966. Flynn's categorization approach is conceptually supported by the idea of an information stream, which acts as the conceptual underpinning. The two different kinds of information that can move through a processor are instructions and data. The instructions that have been delivered to the processing unit are carried out in the order that they were received, and this order is referred to as the instruction stream. The information that is transferred from the memory to the processing unit is referred to as the "data stream," and it is known by this name for a reason. Either the data stream or the instruction stream can be single or multiple at the same time, according to Flynn's classification of the situation. The following is a rundown of the four primary subfields that come together to form computer architecture:

Single-Instruction Single-Data Streams (SISD);
Single-Instruction Multiple-Data Streams (SIMD);
Multiple-Instruction Single-Data Streams (MISD);
Multiple-Instruction Multiple-Data Streams (MIMD)

6.1 Multivector and SIMD Computers

Vector Supercomputers

If the user so wishes, a vector processor can be layered onto the capability of a scalar processor in a vector computer. This is done at the user's discretion. The data and programs are initially loaded into the primary memory of the computer that is serving as the host for the session. Following that, it is the job of the scalar control unit to decode each and every instruction. The scalar processor will use scalar functional pipelines to carry out the operations at hand if the instructions that have been decoded turn out to be scalar operations or program operations. After the instructions have been decoded, a check is performed to determine whether or not they involve vector operations. When it is determined that they are, the instructions are transmitted to the vector control unit.

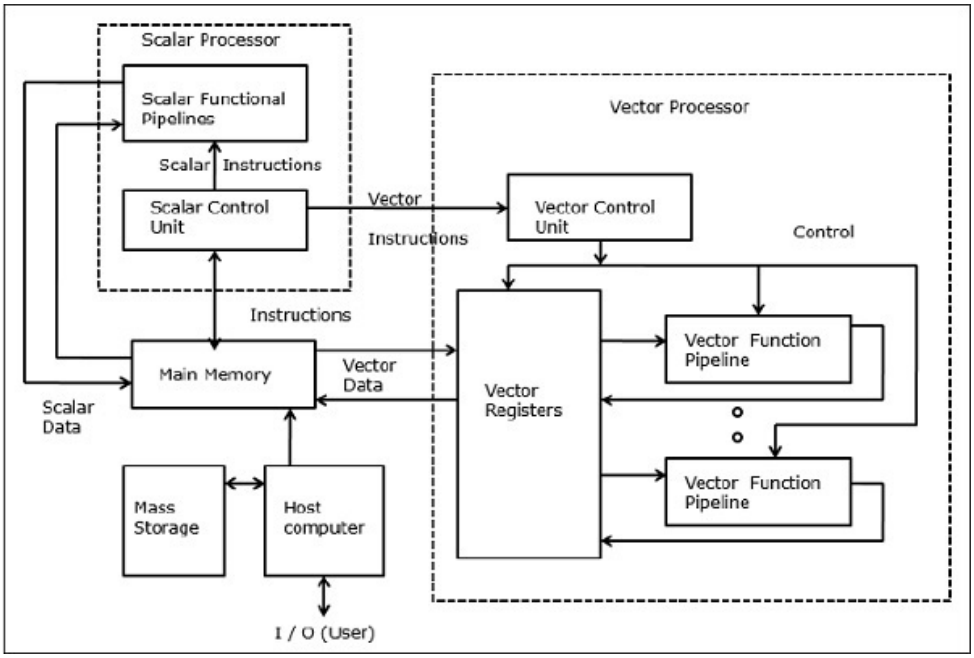

Figure 6: Multivector and SIMD Architecture

6.2 MIMD Architecture Multiple

A vast number of processors and memory modules are connected to one another in a MIMD parallel architecture by means of an interconnection network of some form. These components are what make up the architecture's overall structure. This collection is comprised of two categories: shared memory and message passing. Shared memory is the more fundamental of the two. Processors are able to communicate with one another by utilizing their core shared memory while employing computer systems that make use of shared memory. Processors will use their interconnection network to communicate with one another when using message-passing systems. When referring to a system that makes use of shared memory, "inter-processor coordination" most often refers to the utilization of a global memory that is shared by all of the processors that are present in the system. The majority of the time, these will be server systems that connect with one another by utilizing a bus and a cache memory controller. Because of the architecture of the bus and cache, there is no longer a requirement for costly multi-ported memory and interface equipment. This is because of the bus/cache architecture. A message-passing paradigm is no longer required for the development of application software, which is another need that has been eliminated. These kinds of computers are also referred to as SMP systems, which stand for symmetric multiprocessor systems. This is because shared access memory is symmetrical, which is the reason why this is the case. Each and every processor has an equal opportunity to read and write to memory, as well as an equal access rate to the memory itself.

7. Conclusion

Based on our findings, parallel computation can assist in improving performance, lowering costs, and saving time.

References

[1] Abraham, S. & Padmanabhan, K., 1989, Performance of the direct binary n-cube network for multiprocessors. *IEEE Transactions on Computers*, 38 (7), 1000– 1011.

[2] Agrawal, P., Janakiram, V. & Pathak, G., 1986. Evaluating the performance of multicomputer configurations. *IEEE Transaction on Computers*, 19 (5), 23 – 27.

[3] Almasi, G. & Gottlieb, A. (1989). Highly Parallel Computing. Benjamin Cummings.

[4] Al-Tawil, K., Abd-El-Barr, M. & Ashraf, F., 1997. A survey and comparison of wormhole routing techniques in mesh networks. *IEEE Network*, 38 – 45.

[5] Bhuyan, L. N. (ed.), 1987. Interconnection networks for parallel and distributed processing. *Computer (Special issue)*, 20 (6), 9 – 75.

[6] Bhuyan, L. N., Yang, Q. & Agrawal, D. P., 1989. Performance of multiprocessor interconnection networks. *Computer*, 22 (2), 25 –37.

