Advanced Architectures

Virtualization

The American Heritage Dictionary offers two (2) applicable definitions of **virtual** that describe the usage of the word in modern computing:

- It is an existence or a result in the essence or effect though not in actual fact, form, or name.
- It is created, simulated, or carried on by means of a computer or computer network.

The **virtualization** of a computer allows a single computer to appear as a multiplicity of computers, each with its own operating system and hardware resources. An individual computer system is used to simulate multiple computers, all sharing the same CPU and I/O facilities. The simulated machines are known as **virtual computers** or **virtual machines**.

Parallel Computing

Preemptive multitasking refers to a task in which the operating system uses some criteria to decide how long to allocate to any one task before giving another task a turn to use the operating system. The term "preemption" implies the ability of interrupting a task, switching to another one, and then resuming the first task at a later time. The term "multitasking", on the other hand, refers to the ability to run simultaneous processes in a given time unit.

Processes that run in an operating system is not the only way to perform several operations at the same time. This activity could be done with the use of simultaneous subtasks called **threads**.

From coining preemptive multitasking and parallel computing, there are terms that must be considered:

- **Concurrency** is defined as "the perception of having tasks that run at the same time."
- **Parallelism** are tasks that literally run at the same time.
- Parallelism is a subset of concurrency.

Things run smoothly as long as two (2) or more threads *read* from the same memory location. The troubles kick in when at least one of them *writes* to the shared memory, while the others are reading from it. At this point, two (2) problems can occur:

- **Data Race** – While a writer thread modifies the memory, a reader thread might be reading from it. If the writer has not finished its work yet, the reader will acquire corrupted data.
- **Race Condition** – This occurs when two (2) or more threads can access the same shared data and try to change the data at the same time.

There are some possible solutions for the problems stated above:

- **Synchronization** – It is a way to ensure that resources will be used by only one (1) thread at a time. It is about marking specific parts of the code as "protected" so that two (2) or more concurrent threads will not execute the code simultaneously, which screws up the shared data.
- **Atomic Operations** – A bunch of non-atomic operations can be turned into atomic ones by means of special instructions provided by the operating system. This way the shared data is always kept in a valid state, no matter how other threads access it.
- **Immutable Data** – Shared data is marked as immutable, which means that nothing can change it. This implies that threads are only allowed to read from it, eliminating the root cause.

Multiprocessor and Multicore Systems

Multicore microprocessor is a microprocessor that contains multiple processors ("cores") in a single integrated circuit. Virtually, all microprocessors today in desktops and servers are multicore and are used to run a parallel processing program.

Task-level parallelism (process-level parallelism) is a method of utilizing multiple processors by running independent programs simultaneously. The difficulty with parallelism is not the hardware. It is difficult to write software that uses multiple processors to complete a task faster, and the problem gets worse as the number of processors increases.

Flynn's Taxonomy, which was proposed in 1966, is one of the categorizations of parallel hardware is still used today. It was based on the number of instruction streams and the number of data streams.

- **SISD (Single Instruction stream, Single Data stream [non-parallel computer])**
  - *Single Instruction:* Only one (1) instruction stream is being acted on by the CPU during one (1) clock cycle
  - *Single Data:* Only one (1) data stream is being used as input during one (1) clock cycle
  - Deterministic execution
- **SIMD (Single Instruction stream, Multiple Data streams [parallel computer])**
  - *Single Instruction:* All processing units execute the same instruction at any given clock cycle
  - *Multiple Data:* Each processing unit can operate on a different data element
  - Best suited for specialized problems characterized by a high degree of regularity, such as graphics or image processing.
  - Synchronous (lockstep) and deterministic execution
  - Two (2) varieties: *processor arrays* and *vector pipelines*
- **MISD (Multiple Instruction streams, Single Data streams [parallel computer])**
  - *Multiple Instruction:* Each processing unit operates on the data independently via separate instruction streams
  - *Single Data:* A single data stream is fed into multiple processing units
  - Few (if any) actual examples of this class of parallel computer have ever existed; used in multiple frequency filters operating on a single signal stream or in multiple cryptography algorithm attempts to crack a coded message
- **MIMD (Multiple Instruction streams, Multiple Data streams [parallel computer])**
  - *Multiple Instruction:* Every processor may be executing a different instruction stream
  - *Multiple Data:* Every processor may be working with a different data stream
  - Execution can be synchronous or asynchronous, and deterministic or non-deterministic
  - Currently the most common type of parallel computer (most modern supercomputers fall into this category)
  - Examples: current supercomputers, networked parallel computer clusters and "grids," multi-processor SMP computers, and multi-core PCs

Multithreading

MIMD relies on multiple processes or threads to try to keep many processors busy. Hardware multithreading allows multiple threads to share the functional units of a single processor in an overlapping fashion to try to utilize the hardware resources efficiently. To permit this sharing, the processor must duplicate the independent state of each thread. These are the different kinds of multithreading:

- **Fine-grained multithreading** switches between threads on each instruction, resulting in interleaved execution of multiple threads. This interleaving is often done in a round-robin fashion, skipping any threads that are stalled at that clock cycle.
- **Coarse-grained multithreading** was invented as an alternative to fine-grained multithreading. It switches threads only on expensive stalls, such as last-level cache misses.

- **Simultaneous multithreading (SMT)** is a variation of hardware multithreading that uses the resources of a multiple-issue, dynamically scheduled pipelined processor to exploit thread-level parallelism and, at the same time, exploit instruction-level parallelism. An example of using SMT is hyperthreading. *Hyperthreading* is an Intel-made high-performance computing architecture that simulates some degree of overlap when executing two (2) or more independent sets of instructions. This is a feature of certain Intel chips that makes one (1) physical CPU appear as two (2) logical CPUs.

Vector/Array Processing

An older and more elegant interpretation of SIMD is called **vector architecture**. Its basic philosophy is to collect data elements from memory, put them in order into a large set of registers, operate on them sequentially in registers using *pipelined execution units*, and then write the results back to memory. Its key feature, therefore, is a set of vector registers.

Instruction set architectures, which are called **scalar architectures** in this context, also pertain to a single vector instruction that specifies a great deal of work—it is equivalent to executing an entire loop. The instruction fetches the needed bandwidth and decodes it afterward.

By using a vector instruction, the compiler or programmer indicates that the computation of each result in the same vector is independent of the computation of other results. This is so the hardware does not have to check for data hazards within a vector instruction.

High-Performance Computing (HPC)

It refers to the usage of aggregated computing power for handling computation and data-intensive tasks that include simulation, modeling, and rendering, in which standard workstations are unable to address. One way of addressing HPC is through clustering.

A **cluster** is a group of loosely coupled computers which is configured to work together as a unit. Unlike the tightly coupled multiprocessing system, each computer in a cluster is a complete unit with its own CPU, memory, and I/O facility. Each computer in the cluster is called a **node**. Different nodes of a cluster may be located in the same physical cabinet or located miles apart, provided there is a way to interconnect the high-speed messaging link and, if applicable, the shared-disk links. There are two (2) primary models used for clustering: **shared-nothing** model and **shared disk** model.

Characteristics of Clustering

- To increase the computing power by combining the power of the individual systems
- To create fault-tolerant systems
- To create high-availability systems
- For load-balancing systems with large workloads

The field of high-performance computing, sometimes called **supercomputing**, arose in an attempt to meet the challenge of solving difficult problems that require massive amounts of computing power. There have been different approaches to high-performance computing, but recently developed systems tend to fall loosely into one of these three (3) architectural categories:

- Systems that are built from clusters of powerful machines (Beowulf clusters)
- Cloud systems
- Systems that use the spare processing capacity of computers connected to a network (grid computing).

Beowulf Clusters

These are simple, highly configurable clusters designed to provide high performance at a low cost. These consist of multiple computers connected by a dedicated, private Ethernet which serves as the link between the computers in the cluster. Each node contains a CPU, memory, an Ethernet connection, and, sometimes, hard disks, and other peripherals.

Linux is generally the OS of choice for Beowuld clusters because of its flexibility. In addition to being configurable, this OS provides the tools needed to configure the cluster to include all the features of a powerful distributed system.

Beowulf clusters are generally configured with one of the two (2) types of computer components:

- **Commodity-off-the-shelf**, or **COTS**, components are simply inexpensive computers connected to form a Beowulf cluster.
- **Blade** components, often called **blade servers**, are computers mounted on a board similar to a motherboard that can be plugged into connectors on a rack.

The network connection between the nodes is not accessible from outside of the cluster. This eliminates security concerns other than the authentication required to maintain cluster integrity.

Additionally, Beowulf clusters are ideal to use as Web servers because blades can be added or removed as required to maintain performance levels under varying loads. With their distributed processing capability, these can also be used effectively for shared or parallel processing, where a single large task is divided into subtasks that different computers within the cluster can process simultaneously.

Grid Computing

Gelernter (as cited in Patterson, 2017) proved in his research that it was possible to produce supercomputer performance for processing large problems. This can be done by distributing the problem and using the spare processing time of personal workstations connected to a network. Issues include dividing the workload effectively, scheduling work, preventing interference with local processing, using the results effectively, and security and privacy for client machines.

The processing algorithms allow the data to be broken into tiny chunks for analysis. On a smaller scale, grid computing is moving into several large financial enterprises to provide more processing capability for their employees. This is done by harnessing the combined and unused processing power of their servers and their end-user workplace computers to augment their machines for the fast solutions of large financial applications.

Cloud Computing

It can be viewed as a simple but potentially powerful conceptual expansion of **client–server computing**. Its basic premise is that many functions of an organization's data center can be moved to services on the Internet, which is "in the cloud." In its simplest form, cloud service provides off-site storage facilities for organizations.

This service can be used as a backup resource or for immediate emergency recovery. It can also be used to give users access to files from anywhere the Internet is available, as well as provide additional computing capability when and where it is needed. There are two common usage of cloud computing:

- **Software as a service (SaaS)** provides software applications that run on a server, delivering the results to the display on a client.
- **Infrastructure as a service (IaaS)** offers cloud-based hardware emulation in the form of virtual machines, networking, and the like.

There are many advantages, as well as major risks, to an organization that IT system architects should consider when making decisions about a system design involving services in the cloud.

Advantages of Cloud Computing

- The client's data center is simplified and the cost is reduced.
- Cloud services provide strong support for user collaboration since multiple users can easily access the same software, databases, tools, and data files from the cloud.

- Properly designed (with proper client applications) cloud services can be accessed on a wide variety of client equipment—fixed or mobile and thick or thin—from anywhere the Internet is available.
- A cloud-based system is inherently scalable.
- A cloud-based system can continue providing services and recovery during an emergency.
- Cloud-based services can be useful for short-term projects with intensive computing needs.
- The use of IaaS allows developers to make risky changes to their virtual machine environment without threatening the safety of the production equipment.

Cloud Computing Risks

- The quality of security at the cloud service is critically important. Even the smallest data leak or data theft can compromise a client organization's future.
- Cloud server outages or a loss of connectivity at any point in the link between the client and the cloud service can impede the ability of users to work.
- The client organization is dependent on the long-term commitment and viability of the cloud service.

Graphics Processing Unit (GPU) Architecture

This is a processor optimized for 2D and 3D graphics, video, visual computing, and display. This term was coined in 2000 when the graphics device had become a processor from a standard Video Graphics Array (VGA) controller.

GPU computing is the term used for computing via a parallel programming language and an application programming interface (API), without using the traditional graphics API and graphics pipeline model. This is in contrast to the **General-Purpose computation on GPU (GPGPU)** approach, which involves programming the GPU using a graphics API and graphics pipeline to perform nongraphic tasks.

Compute Unified Device Architecture (CUDA), on the other hand, is a scalable parallel programming model and software platform for the GPU and other parallel processors that allows programmers to bypass GPU's graphics API and graphics interfaces.

APIs have played an important role in the rapid, successful development of GPUs and processors. These are the standard graphics APIs used in rendering and computing:

- **Direct3D** – This is one of the Microsoft DirectX multimedia programming interfaces.
- **OpenGL** – This open standard was originally proposed and defined by Silicon Graphics Incorporated.
- **OpenCL** – This is made by Khronos Group and is used for GPU Memory buffering.
- **Vulkan** – This is a cross-platform computing API used by AMD.

References:
Barney, B. (2019, May 25). *Introduction to parallel computing.* Retrieved from https://computing.llnl.gov/tutorials/parallel_comp/ on July 9, 2019
Berekovic, M., Buchty, R., Hamann, H., Koch, D., & Pionteck, T. (2018). *Architecture of computing systems – ARCS 2018 proceedings*. Switzerland: Springer Nature.
CUDA Zone. (n.d.). In *NVIDIA.* Retrieved from https://developer.nvidia.com/cuda-zone on July 9, 2019
Elahi, A. (2018). *Computer systems - Digital design, fundamentals of computer architecture and assembly language*. Switzerland: Springer Nature.
Englander, Irv. (2014). *The architecture of computer hardware, systems software & networking* (5th ed.). New Jersey: Wiley.
High-Performance Computing (HPC). (2016, March 7). In *Comsol.* Retrieved from https://www.comsol.com/multiphysics/high-performance-computing on July 9, 2019
Hyperthreading. (n.d.). *Definition of: hyperthreading*. (*In PC Magazine Article*). https://www.pcmag.com/encyclopedia/term/44567/hyperthreading on July 9, 2019
Hyper-Threading Technology. (n.d.). In *Intel®.* Retrieved from https://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html on July 9, 2019
Patterson, D. & Hennessy, J. (2017). *Computer organization and design – The hardware/software interface*. Massachusetts: Elsevier.
Ramey, W. (2009). *Languages, APIs and Development Tools for GPU Computing* [PDF file]. Retrieved from https://www.nvidia.com/content/GTC/documents/SC09_Languages_DevTools_Ramey.pdf on July 9, 2019
Techquickie. (2013, September 24). What is Hyper Threading Technology as Fast As Possible [Video file]. Retrieved from https://www.youtube.com/watch?v=wnS50lJicXc on July 9, 2019
Techquickie. (2016, June 3). How Do CPUs Use Multiple Cores? [Video File]. Retrieved from https://www.youtube.com/watch?v=S3I5WNHbnJ0 on July 9, 2019
Triangles. (2019, March 6). *A gentle introduction to multithreading* [Web log post]. Retrieved from https://www.internalpointers.com/post/gentle-introduction-multithreading on July 9, 2019
Vector Processor. (n.d.). In *Techopedia.* Retrieved from https://www.techopedia.com/definition/9907/vector-processor
What is a High-Performance Computer?. (n.d.). In *Software Carpentry Foundation.* Retrieved from https://epcced.github.io/hpc-intro/010-hpc-concepts/ on July 9, 2019
Yadin, A. (2016). *Computer systems architecture*. Florida: CRC Press.