Project A3 Parallel Programming Report

Definitions

1) A *Task* in computing can has several meanings, depending on the operating system. In some systems a task is broadly described as a unit of work. It can be considered as a request to perform a certain action, or it can be the request and the action being processed by the processor in the case of *threads*. In LINUX a task is considered as a unit of execution.
2) *Pipelining* is a type of parallel computing that breaks a task into steps performed by different processors in the computer.
3) *Shared Memory* exists in two dimensions: hardware, and programming. In the hardware dimension shared memory describes when all processors are physically bussed to a common memory. From a programming perspective, shared memory is a process in parallel programming where all of the tasks/processes access the same memory address that they can read and write to at the same time. Open MP exploits the shared memory process.
4) *Communication* in parallel programming is simply the process of processors exchanging data and interacting during tasks. There is a formal language that describes the patterns of interactions in concurrent system called Communicating Sequential Processes (CSP).
5) *Synchronization* is a concept often associated with communication where two tasks are synchronized by a clock or a signal encoded into a data stream. Often in the case of two synchronized tasks that when one task is complete it must wait for the other task to complete (or reach a certain point) before the first task can move on.

Flynn's Taxonomy

Flynn's Taxonomy classifies multiprocessor computers based on two factors: data stream, and instruction streams. The difference in processors is in whether they are single or multi instructions and/or data. The classifications are so named.
Single Instruction Single Data (SISD) computers are classified as such, because they send one data stream to the CPU during one clock cycle, and the CPU acts on only one instruction during one clock cycle. They are called sequential computers. Computers that employ SISD are the oldest type of computer, and what we call single core PC's.
Single Instruction Multi Data (SIMD) computers are the most basic type of parallel computers. In this one the CPU again operates on only one instruction during one clock cycle, but it sends multiple data streams during one clock cycle. SIMD is best employed in solutions involving images and graphics. Most modern computers take this approach to computing.

Multiple Instruction Single Data (MISD) computers can send multiple instructions in one clock cycle, but read only one data stream during one clock cycle. These computers operate only in theory. There has never been one to actually be employed.
Multiple Instruction Multiple Data (MIMD) computers are the most common type of parallel computers today. With MIMD each processor can execute multiple instructions, while reading multiple data streams in one clock cycle. Each execution can be synchronous, deterministic, or not.

Parallel Programming Models

Parallel Programming Modelling is an abstraction, or generalization of parallel architecture. As stated in the Introduction to Parallel Programming, there are several *Parallel Programming Models.* The models listed in our guide are:
Shared Memory (without threads)*, Threads, Distributed Memory/ Message Passing, Data Parallel, Hybrid, Single Program Multiple Data (SPMD), and Multiple Program Multiple Data (MPMD).*
Parallel programming models can be grouped into two categories:
Process interaction, in which models like shared memory exist. Shared memory is also called "virtual shared memory". It appears to the user to have one global memory address, but it is actually physically distributed across many network machines.
The message passing model is also a process interaction approach to parallel programming. We can refer back to our definition of communication with this one, where processes exchange data through passing messages to one another.
Problem decomposition, which focuses on threads of execution, where instructions are processed through a single CPU.

Parallel Computer Memory Architecture

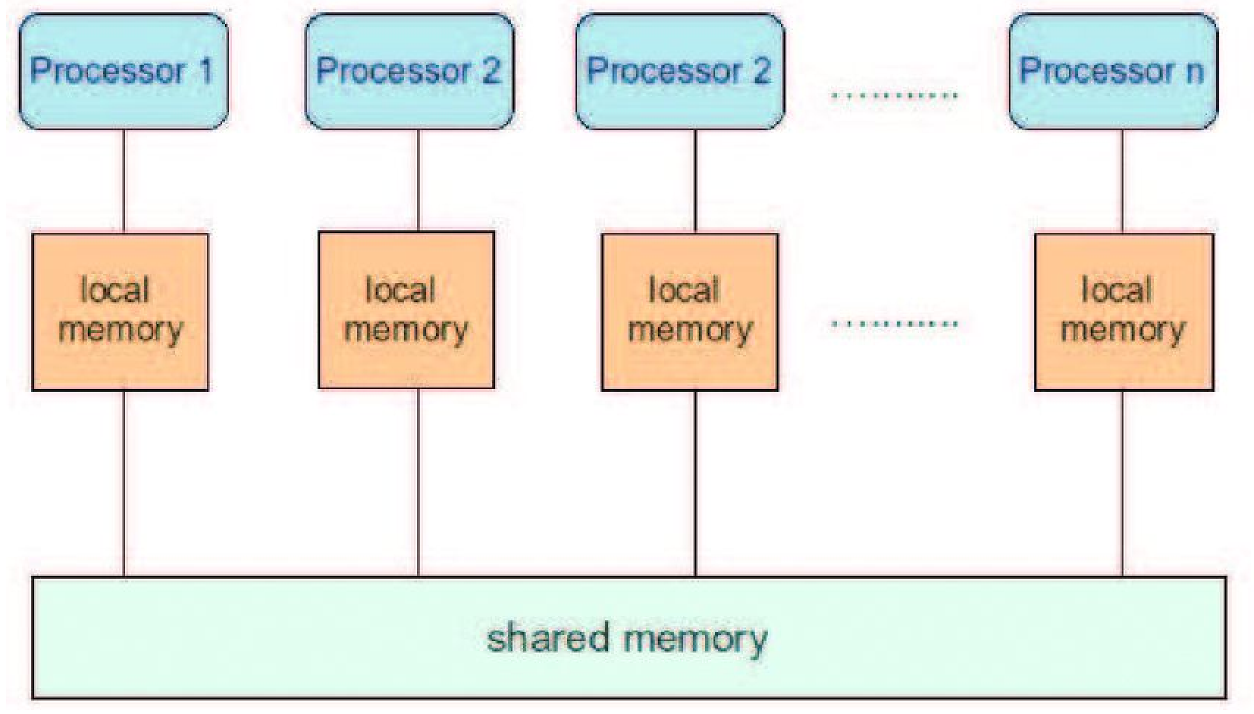
In the *Shared Memory* architecture we know that all the processors share a global memory address. According to our literature shared memory machines are classified by two categories: **Uniform Memory Access (UMA)** and **Non-Uniform Memory Access.**
With *UMA*, each machine has identical processors that share equal access and equal access times to the memory. We sometimes refer to this system as *Cache Coherent UMA (CC-UMA)*, meaning if one processor updates a shared memory address, all other processors know about the update. This is achieved on the hardware level. Because all the processors have equal access to all the devices, the system is called a *symmetric multiprocessor (SMP)*.
The other classification of shared memory architecture, *NUMA* is made by linking two or more symmetric multiprocessors, where one SMP can access the memory of another SMP. In this case all memory access to all the machines is not equal. This makes the memory access slower.

However, these machines can still maintain Cache Coherency, and we call this option CC-NUMA.
Open MP uses Non-Uniform Multiprocessor Architecture, because this system makes it possible for OpenMP to implement Hybrid Parallel Programming. This is when OpenMP is combined with a *Message Passing Interface (MPI)* for the distributed memory parallelism. This is referred to as *Hybrid Parallel Programming*. We need this with OpenMP, because by itself, it's parallelism is limited to one node, which is no good for *High Performance Computing* applications.

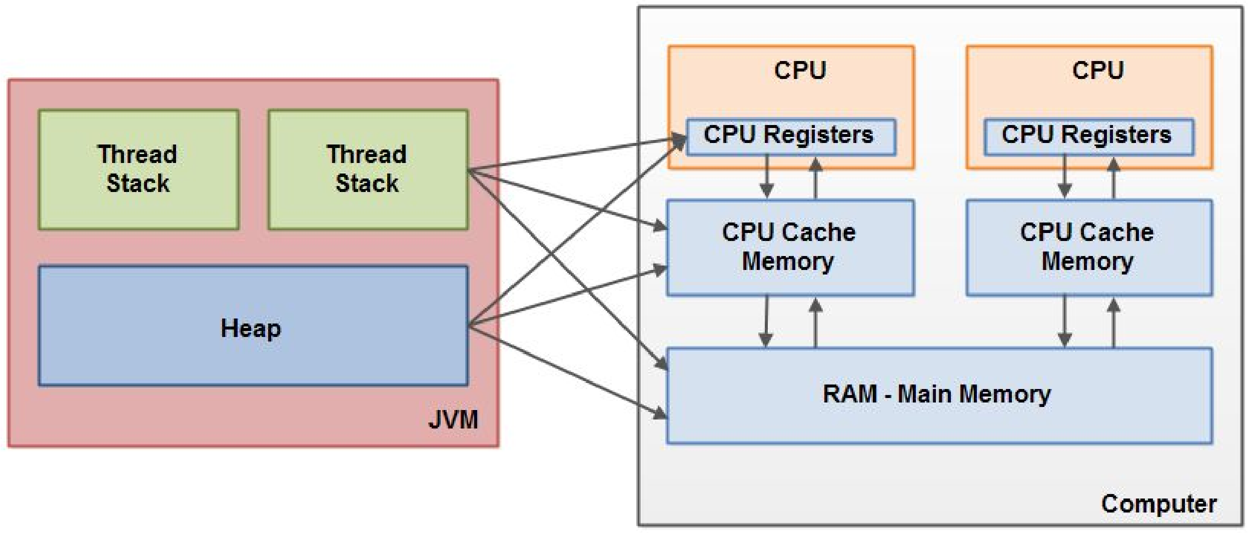
Compare Shared Memory Model and Threads Model

To me it seems hard to compare Shared Memory Model of parallelism with thread model, being that the Thread Model is actually a form of the Shared Memory Model. However, one difference is that Shared Memory Model is an asynchronous system, whereas Threads can execute any subroutine at the same time as other threads, because it employs synchronization constructs to ensure that memory addresses are not called at the same time (which causes race conditions and deadlocks).

Shared Memory Model



Thread Model



What is Parallel Programming

In very simple terms, it is the use of multiple processors, to solve a computational problem. This type of programming takes a problem, breaks it down into a series of smaller steps, delivers instructions, and processors execute the solutions at the same time. It is also a form of programming that offers the same results as serial programming but in less time and with more efficiency.

What is System on Chip (SoC)

System on Chip is an integrated circuit that has all of the components of a computer on it. The Rasberry Pi uses a System on Chip circuitry.

Advantages of SoC

Some of the advantages of System on Chip architecture are:

1) The computer is smaller, taking up less space.
2) It's much less expensive than a traditional PC or laptop which houses a separate CPU.
3) It needs lower power input.
4) It has better system reliability.

Project A3 Parallel Programming Report

Definitions

1) A *Task* in computing can has several meanings, depending on the operating system. In some systems a task is broadly described as a unit of work. It can be considered as a request to perform a certain action, or it can be the request and the action being processed by the processor in the case of *threads*. In LINUX a task is considered as a unit of execution.
2) *Pipelining* is a type of parallel computing that breaks a task into steps performed by different processors in the computer.
3) *Shared Memory* exists in two dimensions: hardware, and programming. In the hardware dimension shared memory describes when all processors are physically bussed to a common memory. From a programming perspective, shared memory is a process in parallel programming where all of the tasks/processes access the same memory address that they can read and write to at the same time. Open MP exploits the shared memory process.
4) *Communication* in parallel programming is simply the process of processors exchanging data and interacting during tasks. There is a formal language that describes the patterns of interactions in concurrent system called Communicating Sequential Processes (CSP).
5) *Synchronization* is a concept often associated with communication where two tasks are synchronized by a clock or a signal encoded into a data stream. Often in the case of two synchronized tasks that when one task is complete it must wait for the other task to complete (or reach a certain point) before the first task can move on.

Flynn's Taxonomy

Flynn's Taxonomy classifies multiprocessor computers based on two factors: data stream, and instruction streams. The difference in processors is in whether they are single or multi instructions and/or data. The classifications are so named.

Single Instruction Single Data (SISD) computers are classified as such, because they send one data stream to the CPU during one clock cycle, and the CPU acts on only one instruction during one clock cycle. They are called sequential computers. Computers that employ SISD are the oldest type of computer, and what we call single core PC's.
Single Instruction Multi Data (SIMD) computers are the most basic type of parallel computers. In this one the CPU again operates on only one instruction during one clock cycle, but it sends multiple data streams during one clock cycle. SIMD is best employed in solutions involving images and graphics. Most modern computers take this approach to computing.
Multiple Instruction Single Data (MISD) computers can send multiple instructions in one clock cycle, but read only one data stream during one clock cycle. These computers operate only in theory. There has never been one to actually be employed.
Multiple Instruction Multiple Data (MIMD) computers are the most common type of parallel computers today. With MIMD each processor can execute multiple instructions, while reading multiple data streams in one clock cycle. Each execution can be synchronous, deterministic, or not.

Parallel Programming Models

Parallel Programming Modelling is an abstraction, or generalization of parallel architecture. As stated in the Introduction to Parallel Programming, there are several *Parallel Programming Models.* The models listed in our guide are:
Shared Memory (without threads)*, Threads, Distributed Memory/ Message Passing, Data Parallel, Hybrid, Single Program Multiple Data (SPMD), and Multiple Program Multiple Data (MPMD).*
Parallel programming models can be grouped into two categories:
Process interaction, in which models like shared memory exist. Shared memory is also called "virtual shared memory". It appears to the user to have one global memory address, but it is actually physically distributed across many network machines.
The message passing model is also a process interaction approach to parallel programming. We can refer back to our definition of communication with this one, where processes exchange data through passing messages to one another.
Problem decomposition, which focuses on threads of execution, where instructions are processed through a single CPU.

Parallel Computer Memory Architecture

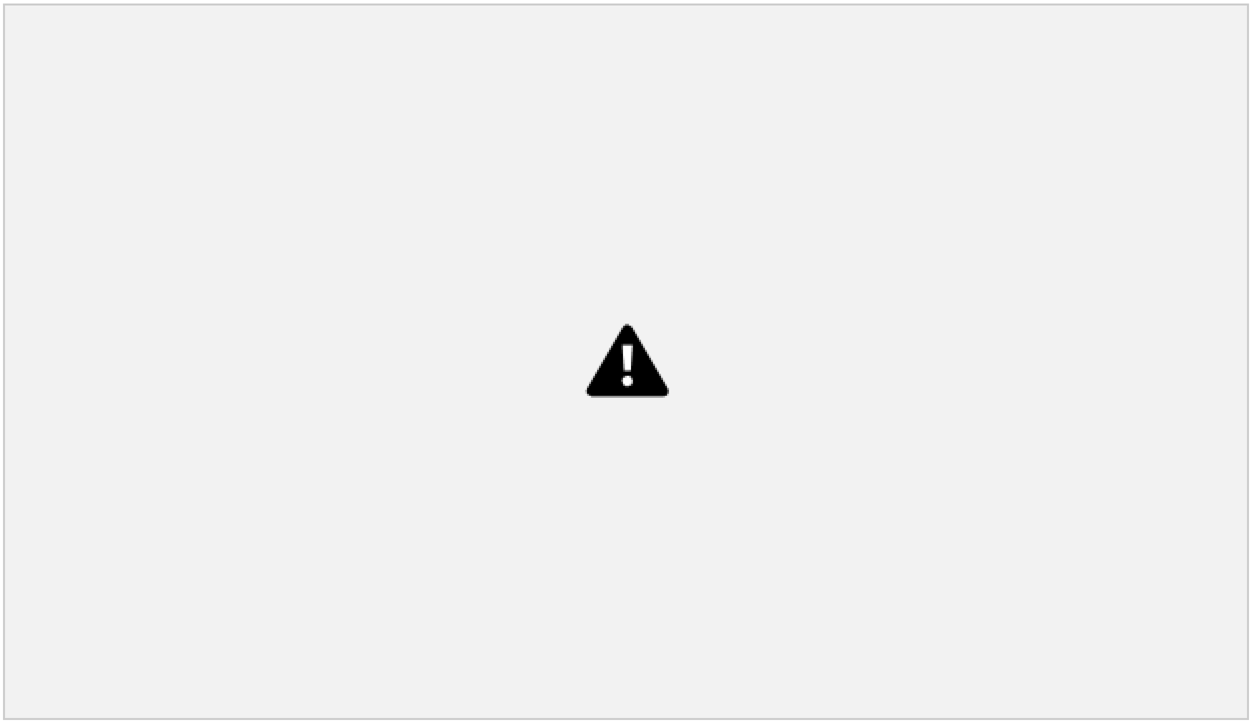
In the *Shared Memory* architecture we know that all the processors share a global memory address. According to our literature shared memory machines are classified by two categories: **Uniform Memory Access (UMA)** and **Non-Uniform Memory Access.**

With *UMA*, each machine has identical processors that share equal access and equal access times to the memory. We sometimes refer to this system as *Cache Coherent UMA (CC-UMA)*, meaning if one processor updates a shared memory address, all other processors know about the update. This is achieved on the hardware level. Because all the processors have equal access to all the devices, the system is called a *symmetric multiprocessor (SMP)*.
The other classification of shared memory architecture, *NUMA* is made by linking two or more symmetric multiprocessors, where one SMP can access the memory of another SMP. In this case all memory access to all the machines is not equal. This makes the memory access slower. However, these machines can still maintain Cache Coherency, and we call this option CC-NUMA.
Open MP uses Non-Uniform Multiprocessor Architecture, because this system makes it possible for OpenMP to implement Hybrid Parallel Programming. This is when OpenMP is combined with a *Message Passing Interface (MPI)* for the distributed memory parallelism. This is referred to as *Hybrid Parallel Programming*. We need this with OpenMP, because by itself, it's parallelism is limited to one node, which is no good for *High Performance Computing* applications.

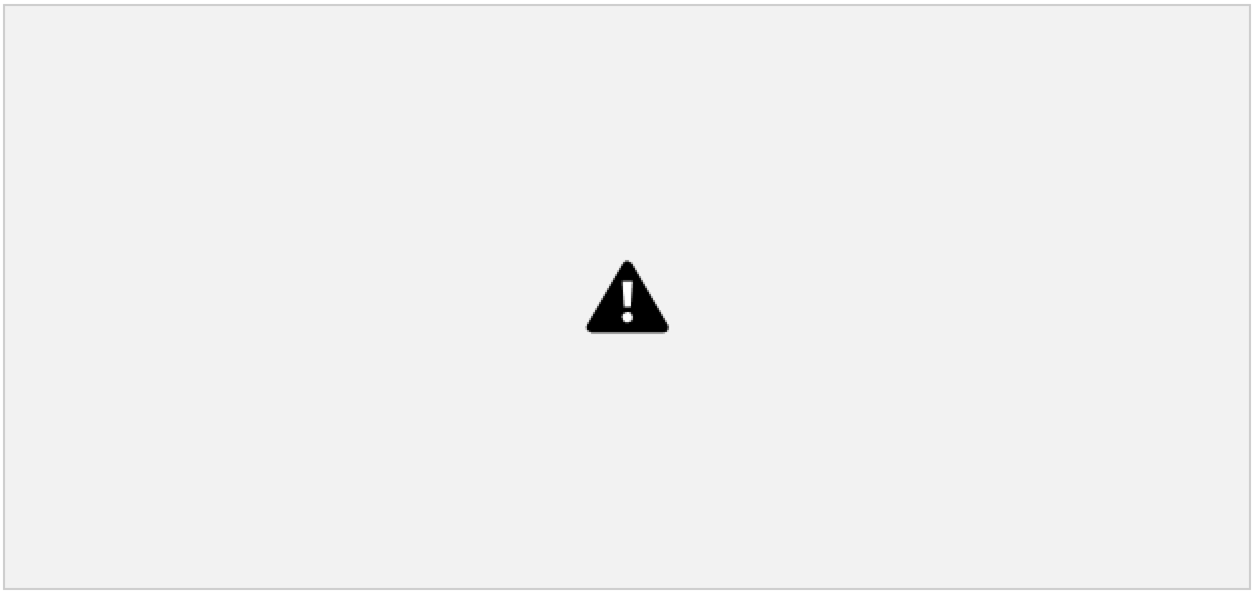
Compare Shared Memory Model and Threads Model

To me it seems hard to compare Shared Memory Model of parallelism with thread model, being that the Thread Model is actually a form of the Shared Memory Model. However, one difference is that Shared Memory Model is an asynchronous system, whereas Threads can execute any subroutine at the same time as other threads, because it employs synchronization constructs to ensure that memory addresses are not called at the same time (which causes race conditions and deadlocks).

Shared Memory Model



Thread Model



What is Parallel Programming

In very simple terms, it is the use of multiple processors, to solve a computational problem. This type of programming takes a problem, breaks it down into a series of smaller steps, delivers instructions, and processors execute the solutions at the same time. It is also a form of programming that offers the same results as serial programming but in less time and with more efficiency.

What is System on Chip (SoC)

System on Chip is an integrated circuit that has all of the components of a computer on it. The Rasberry Pi uses a System on Chip circuitry.

Advantages of SoC

Some of the advantages of System on Chip architecture are:

1) The computer is smaller, taking up less space.
2) It's much less expensive than a traditional PC or laptop which houses a separate CPU.
3) It needs lower power input.
4) It has better system reliability.