LESSON 1

Background

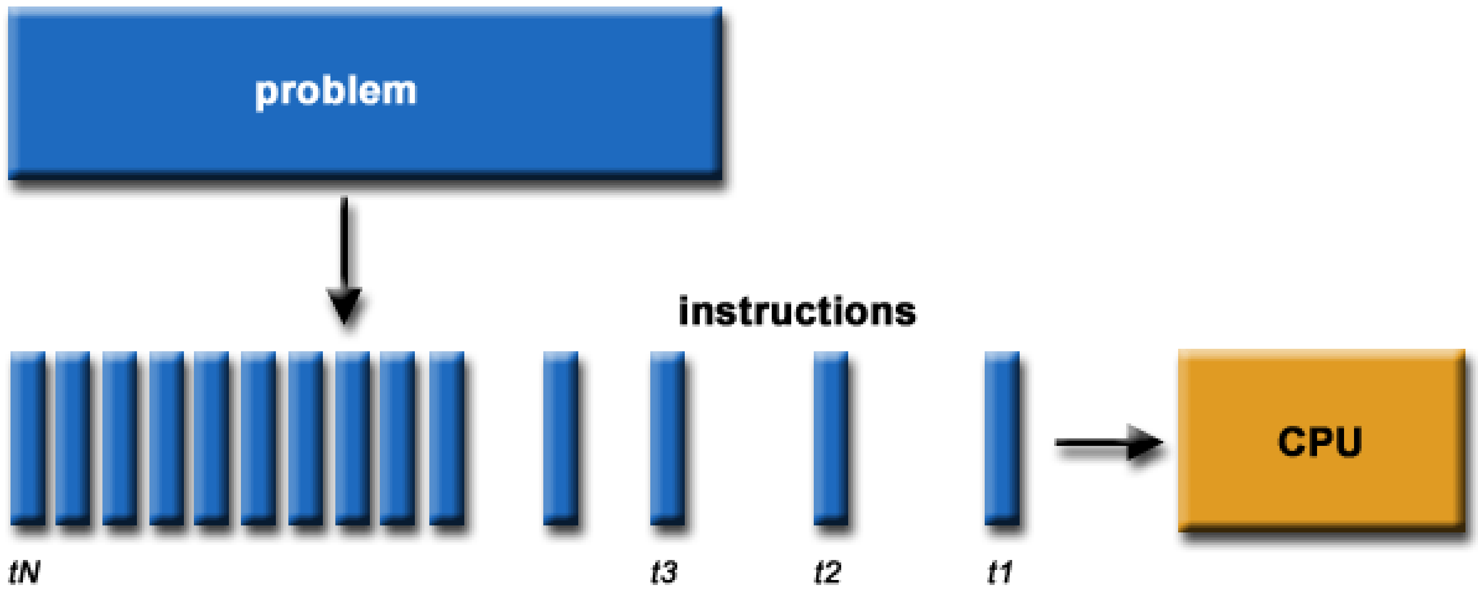
TOPICS

- Serial computing
- Parallel computing
- Programming models

Key words: *SISD, SIMD, MISD, MIMD, SIMT, share memory model, threads model, message passing model, domain and functional decomposition.*

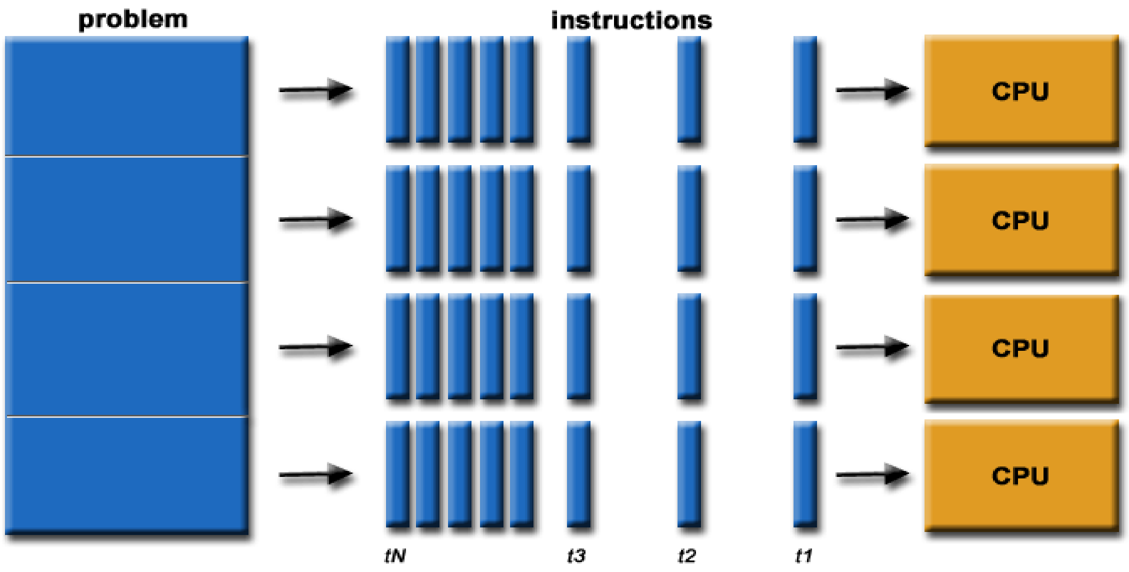
Serial Computing

- Consists in the use of one resource to solve a computational problem:
  - Run on a single computer having a single Central Processing Unit (CPU)
  - The problem is broken into a discrete series of instructions.
  - Instructions are executed one after another.
  - Only one instruction may execute at any moment in time.



Parallel Computing

- Is the simultaneous use of multiple compute resources to solve a computational problem:
  - Run using multiple CPUs
  - The problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
  - Instructions from each part execute simultaneously on different CPUs



- The compute resources might be:
  - A single computer with multiple processors
  - An arbitrary number of computers connected by a network
  - A combination of both

- The computational problem should be able to:
  - Be broken apart into discrete pieces of work that can be solved simultaneously
  - Execute multiple program instructions at any moment in time
  - Be solved in less time with multiple compute resources than with a single compute resource

Flynn's Classical Taxonomy

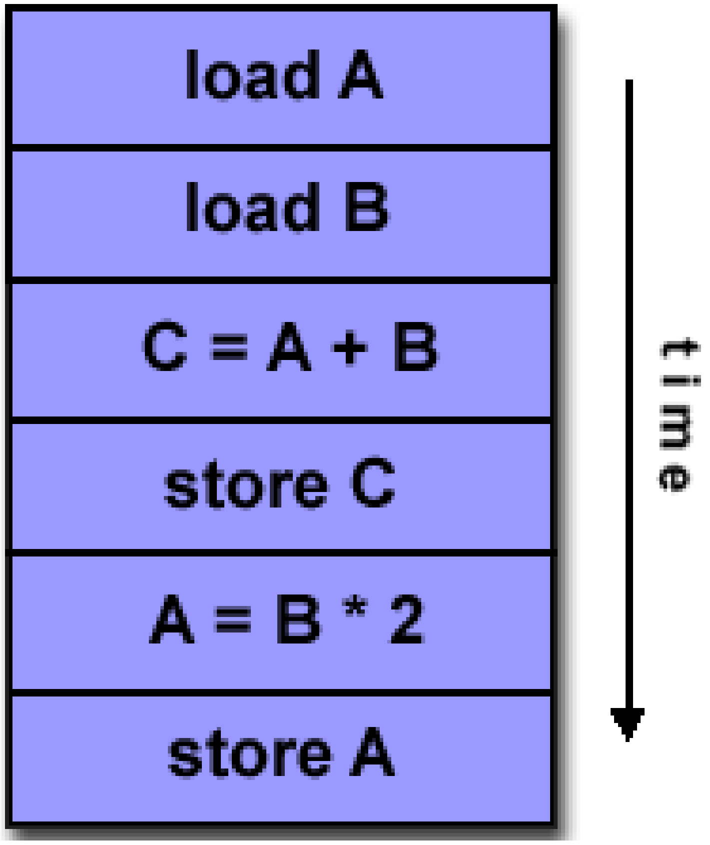
- Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of *Instruction* and *Data*.

- Each of these dimensions can have only one of two possible states: *Single* or *Multiple*.

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

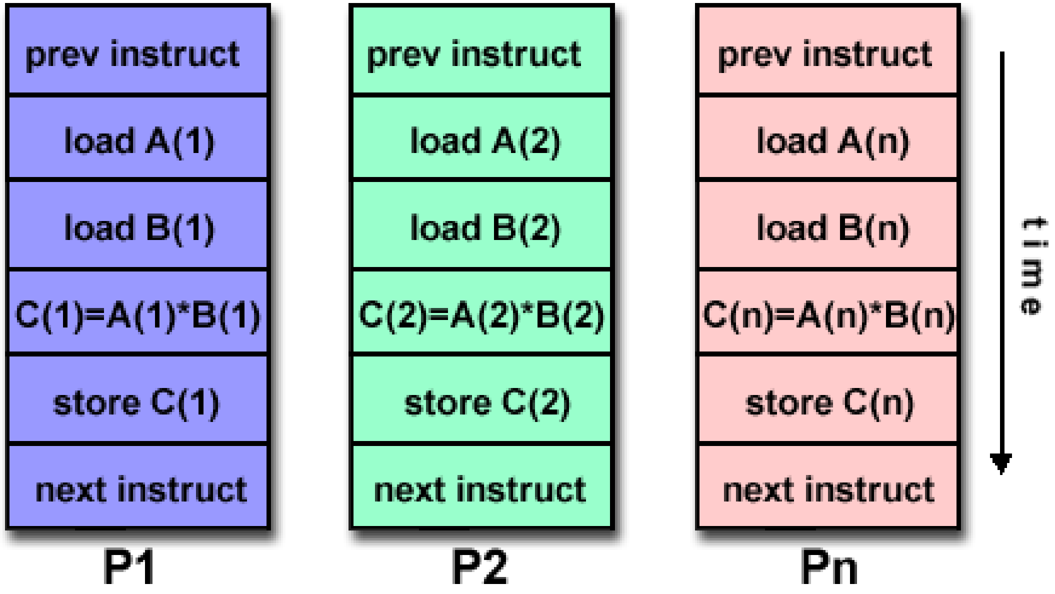
Single Instruction, Single Data (SISD)

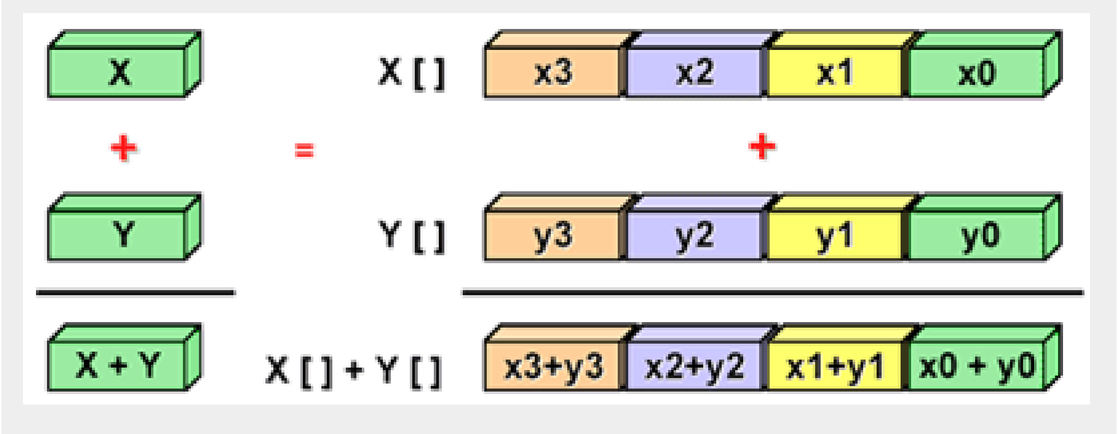
- A serial (non-parallel) computer
- Single Instruction: Only one instruction stream is being acted on by the CPU during any one clock cycle
- Single Data: Only one data stream is being used as input during any one clock cycle



Single Instruction, Multiple Data (SIMD)

- A type of 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



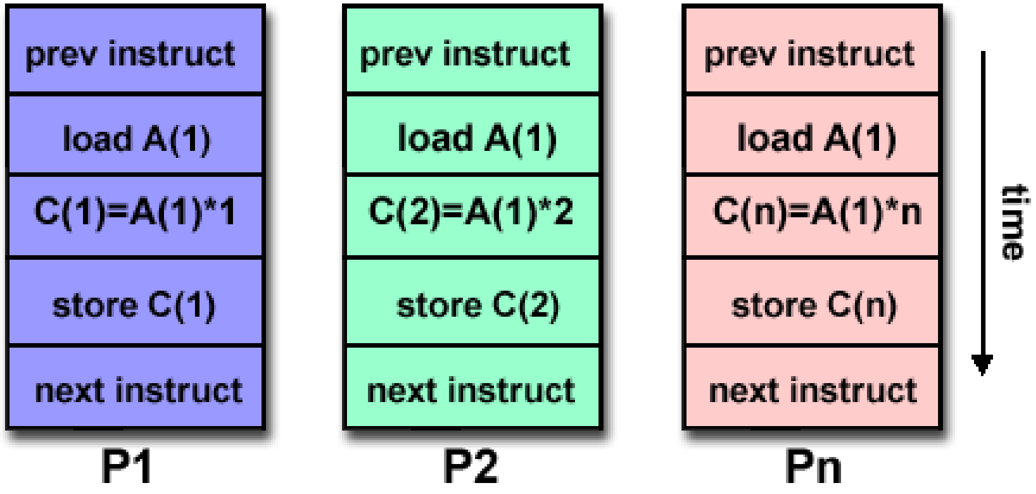


Multiple Instruction, Single Data (MISD)

- A type of parallel computer

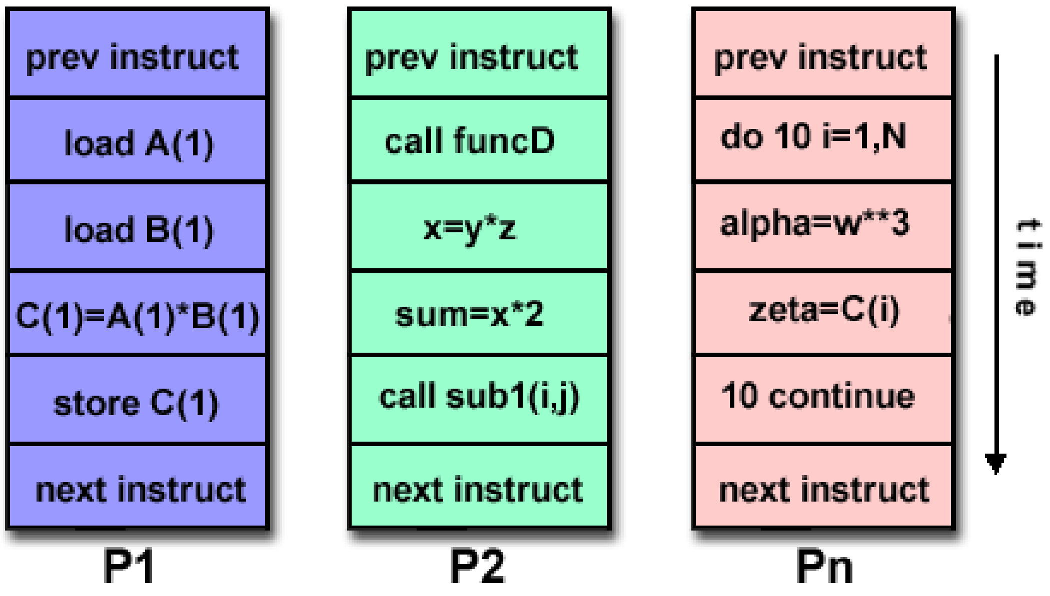
- Multiple Instruction: Each processing unit operates on the data independently via separate instruction streams.

- Single Data: A single data stream is fed into multiple processing units.



Multiple Instruction, Multiple Data (MIMD)

- A type of 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



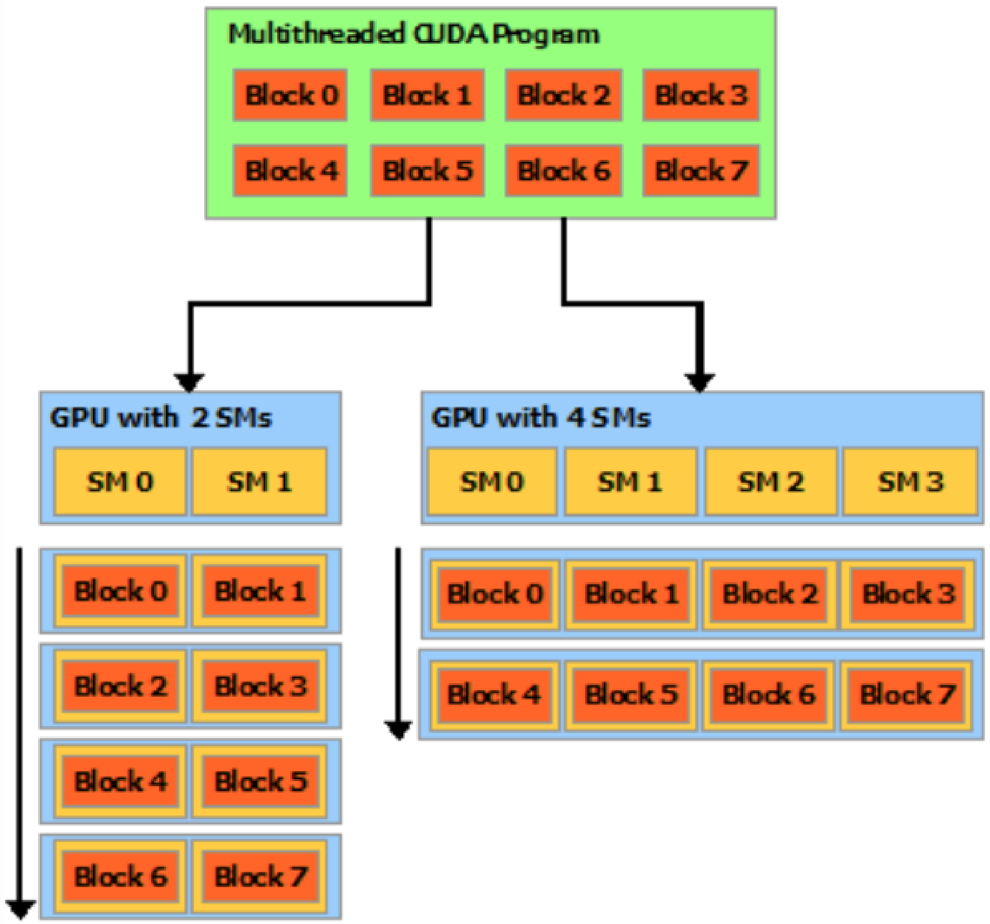
Single Instruction, Multiple Threads (SIMT)

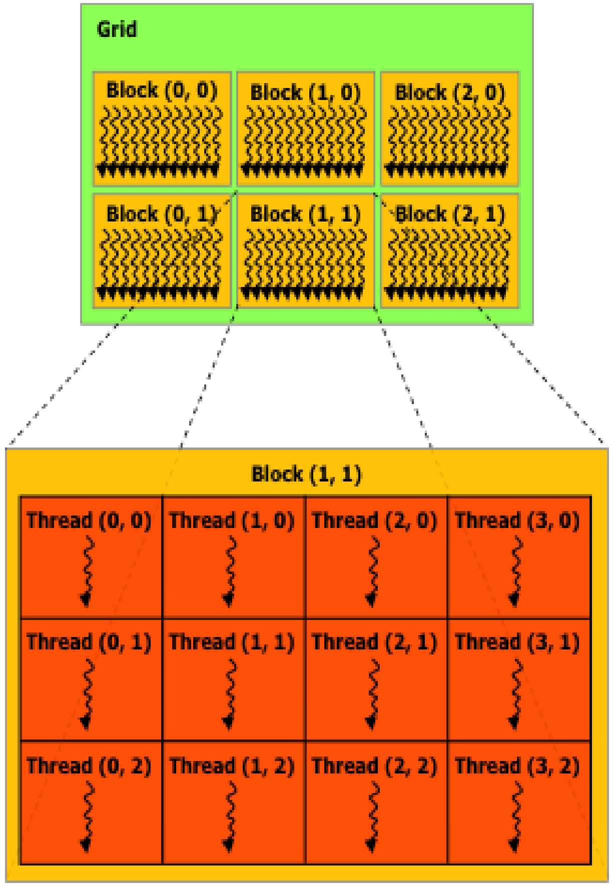
- A GPU is built around an array of Streaming Multiprocessors (SMs) . A multiprocessor is designed to execute hundreds of threads concurrently.

- A CUDA program is partitioned into blocks of threads that execute independently from each other, so that a GPU with more multiprocessors will automatically execute the program in less time than a GPU with fewer multiprocessors.

- To manage such a large amount of threads, it employs a unique architecture called *SIMT* (*Single-Instruction, Multiple-Thread)*.

- The SIMT architecture is akin to SIMD (Single Instruction, Multiple Data) vector organizations in that a single instruction controls multiple processing elements.





Parallel Programming Models

- Shared Memory Model

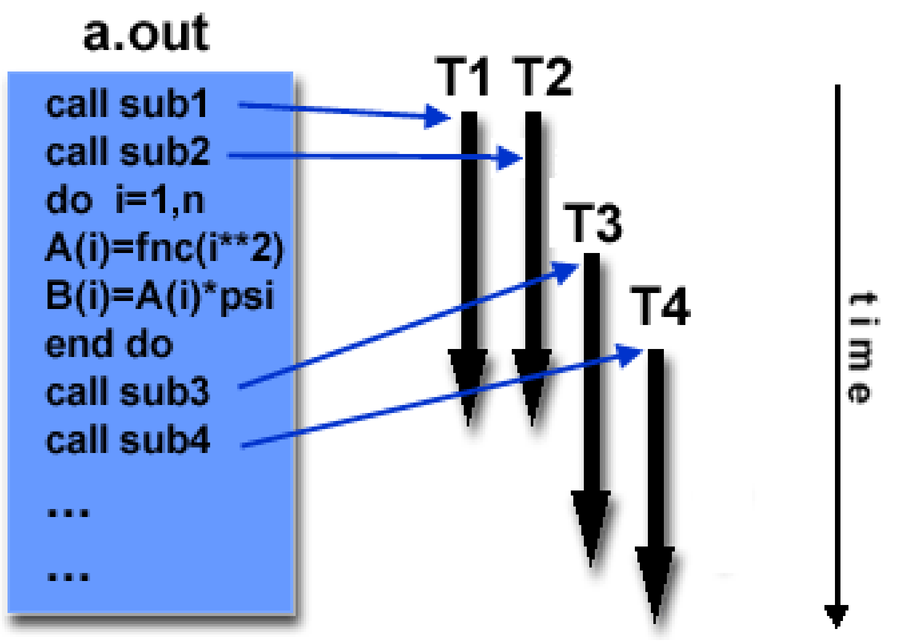
- In this programming model, tasks share a common address space, which they read and write to asynchronously.

- Various mechanisms such as locks / semaphores may be used to control access to the shared memory.

- Threads Model

- This programming model is a type of shared memory programming.

- In the threads model of parallel programming, a single process can have multiple, concurrent execution paths.

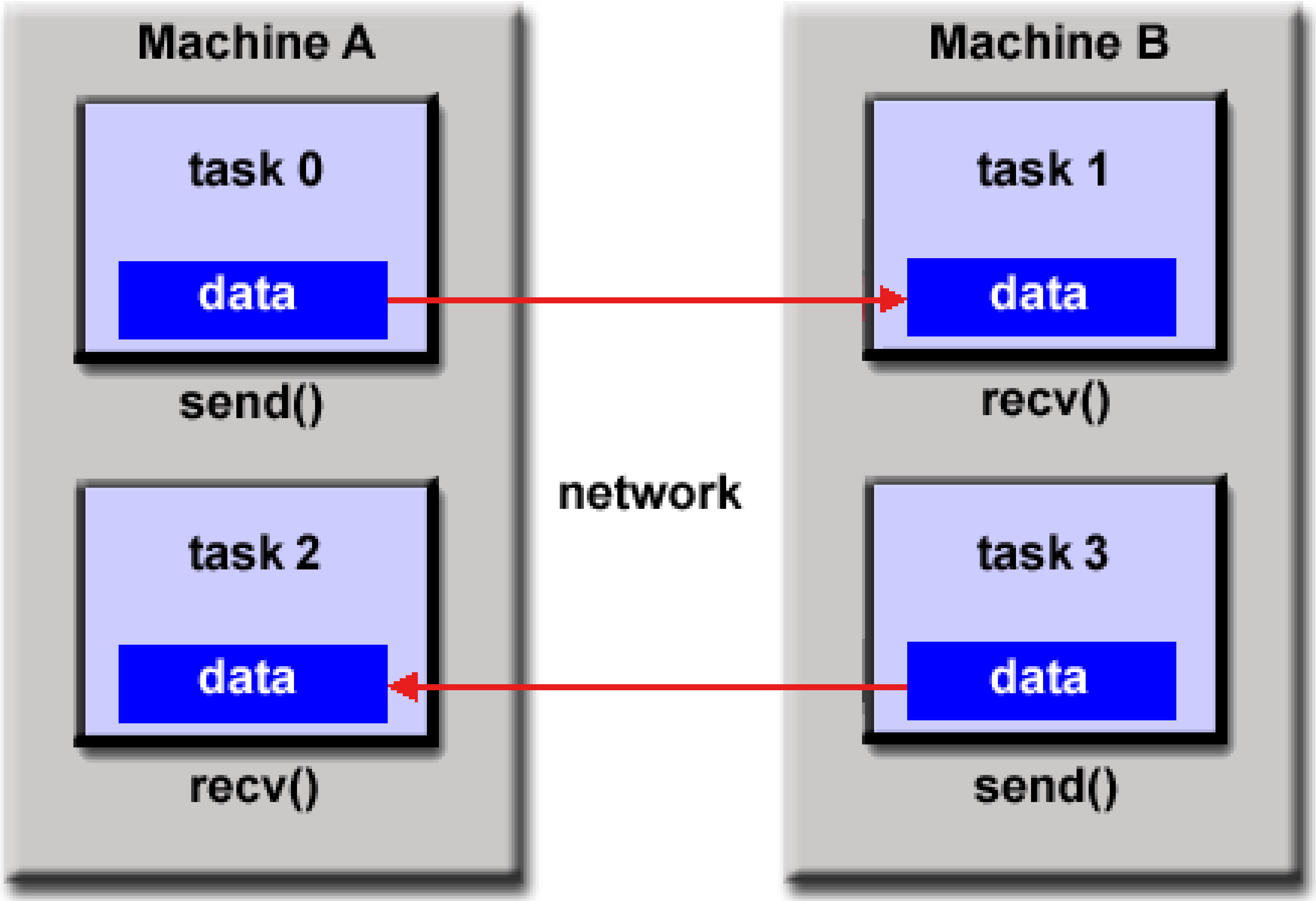


- Distributed Memory / Message Passing Model

- A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines.

- Tasks exchange data through communications by sending and receiving messages.

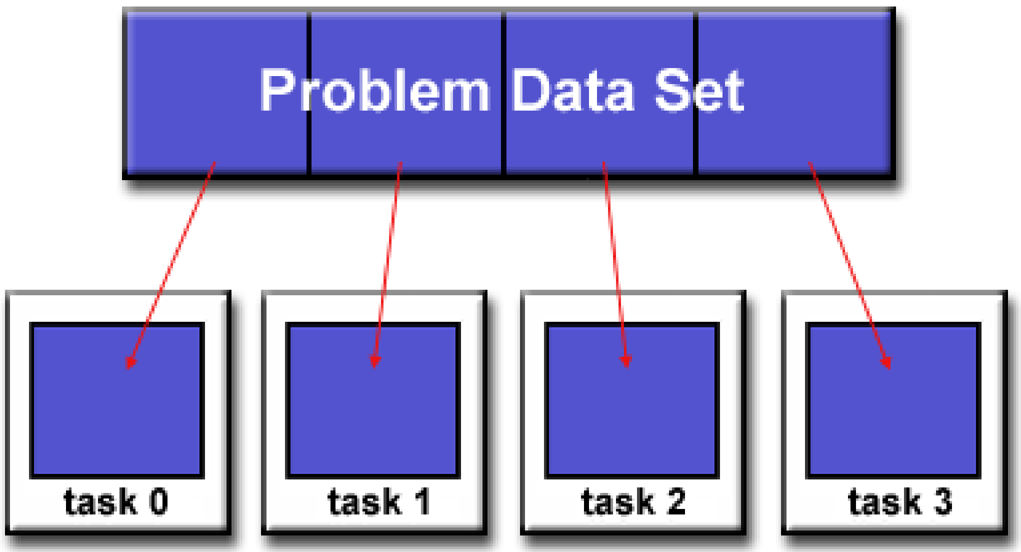
- Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.


Machine A
task 0
data
send()
task 2
data
recv()
network
Machine B
task 1
data
recv()
task 3
data
send()

Designing Parallel Programs

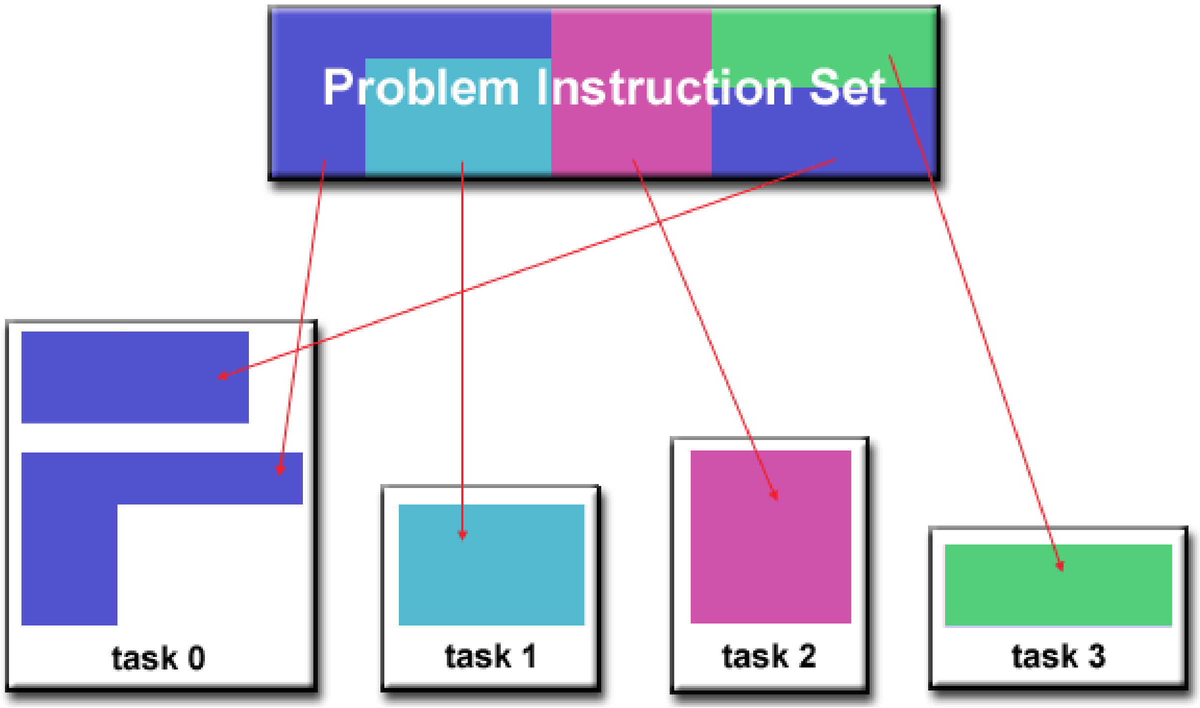
- There are two basic ways to partition computational work among parallel tasks: *domain decomposition* and *functional decomposition*.

- Domain Decomposition: In this type of partitioning, the data associated with a problem is decomposed. Each parallel task then works on a portion of of the data.



- Functional Decomposition: In this approach, the focus is on the computation that is to be performed rather than on the data manipulated by the computation.

- The problem is decomposed according to the work that must be done. Each task then performs a portion of the overall work.



Practice

- Complete the exercise 00add.cu in order to add two vectors. It is a requirement the use of dynamic memory.