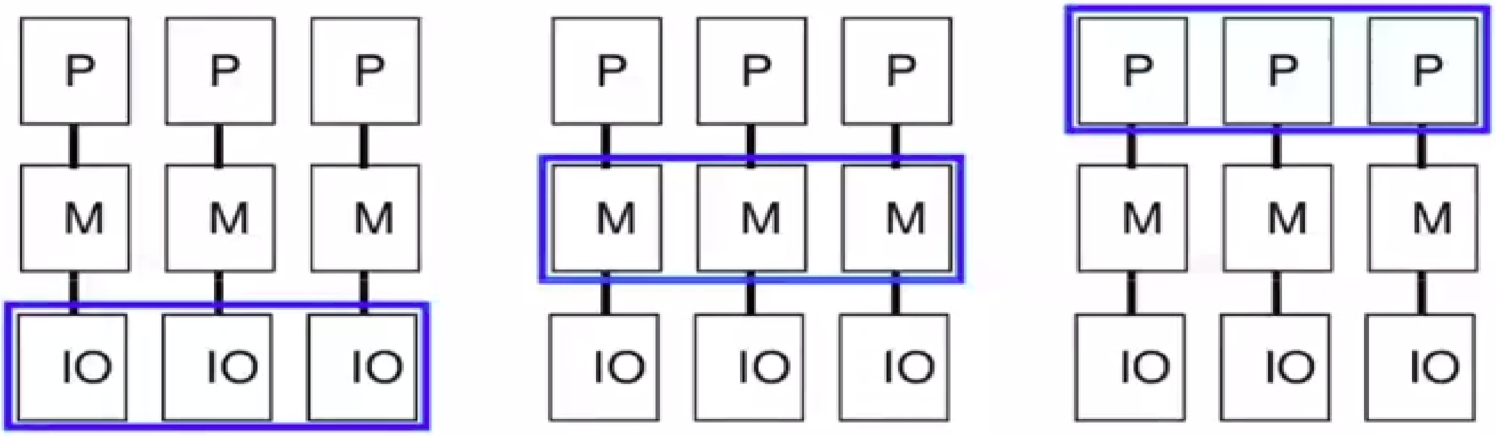
Programming Model

- Provides a communication abstraction that is a contract between hardware and software (a la ISA)
  - Programming model != programming language
- Conceptualisation of the machine that the programmer uses in coding applications
  - How parts cooperate and coordinate their activities
  - Specifies communication and synchronisation operations
- Multiprogramming
  - No communication or synchronisation at program level
- Shared address space
  - Like a bulletin board
- Message Passing
  - Like letters or phone calls, explicit point of contact
- Data Parallel
  - More regimented, global actions on data
  - Implemented using either shared address space or message passing

Historical View

- **Historically: system architecture and programming model were tied together**



Join At:	I/O (Network)	Memory	Processor
Program With:	Message Passing	Shared Memory	Data Parallel
			Single-Instruction Multiple-Data (SIMD)

Historical view...

- Architecture -> Programming Model
  - Join at network->program with message passingmodel
  - Join at memory->program with shared memory model
  - Join at processor->program with SIMD or data parallel
- Programming Model ->Architecture
  - Message-passing programs on message-passing arch
  - Shared-memory programs on shared-memory arch
  - SIMD/data-parallel programs on SIMD/data-parallel arch
- But
  - Isn’t hardware basically the same? Processors, memory,&I/O?
  - Convergence! Why not have generic parallel machine & program with model that fits the problem?

Coordination

- When the cores can work independently, writing a parallel program is much same as writing a serial program

  It gets complex when the cores need to coordinate their work

- Coordinationinvolves
  - Communication
    - Send partial sums to another core
  - Load Balancing
    - We want each core to do same amount of work. Otherwise some cores are working and others are waiting idle wasting computational power
  - Synchronisation
    - Master reads the values in the array
    - Then cores must start computing. Else wait
    - Add a point of synchronisation between read value and compute partial sums

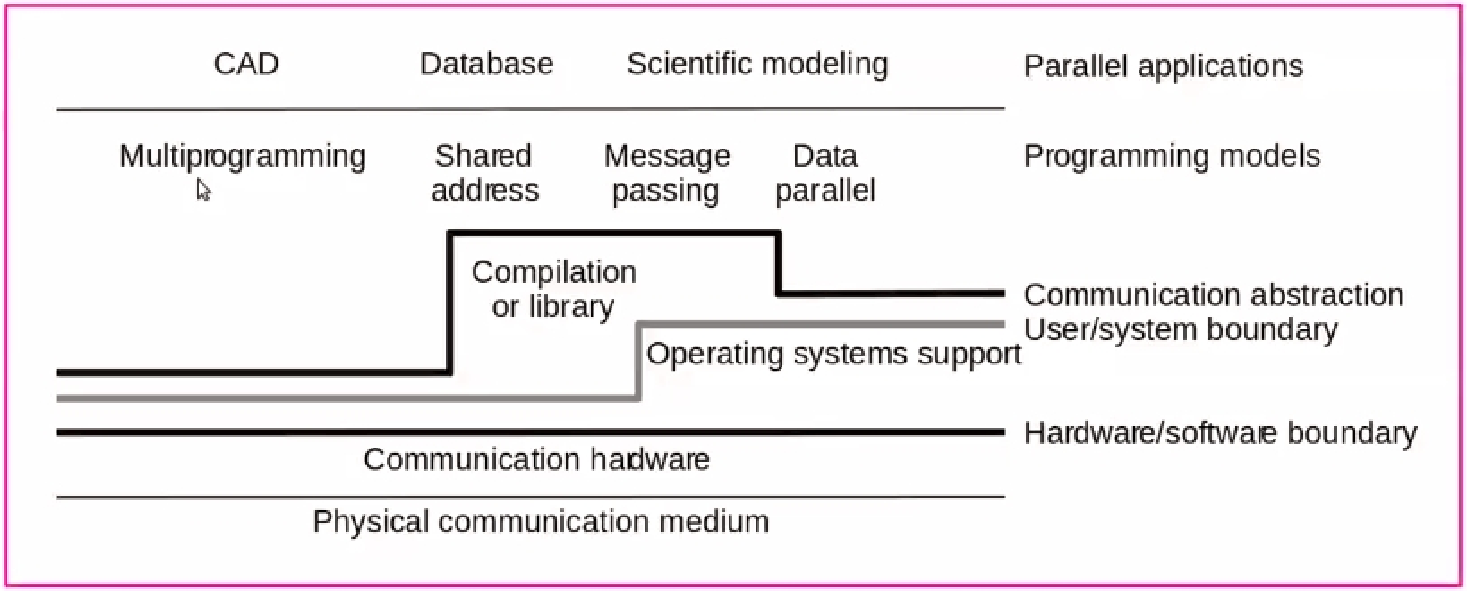
Parallel Programming

- For Distributed Memory and Shared Memory Systems
- Using C and its extensions
- MPI: Message Passing Interface
  - For Distributed memory systems
  - Libraries of type definitions + functions+macros
- POSIX threads: pthreads
  - For shared memory systems
  - Libraries of type definitions + functions+macros
- OpenMP
  - For shared memory systems
  - Library+modifications to C compiler

Parallel Architectures

- Almasi and Gottlieb 1989
  - "Parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast"

Layers of Abstraction



As programming models have become better understood and Implementation techniques have matured, compilers and run-time libraries have grown to provide an important bridge between the programming model and the underlying hardware

Layers of abstraction in parallel computer architecture

- The framework to understand communication in parallel machines is shown in the figure (another slide)
- Top layer = programming model
  - Specifies how the programs running in parallel communicate information to one another, and
  - What synchronisation operations are available to coordinate activities
- Communication abstraction = user level communication primitives
  - Provided directly by hardware,or
  - By operating system,or
  - Machine specific user software