# Introduction to Parallel Architectures

Instructor: Josep Torrellas

CS533

# Introduction

- Why parallel architectures:
  - Absolute performance
  - Power and energy
  - Complexity
  - Cost-performance
- Key enabling factors:
  - Advances in semiconductor and interconnect technology
  - Advances in software technology

# Classification of Parallel Machines

- Michael Flynn's classification
- Model-based classification

# Flynn's Classification

- Based on the notions of:
  - Instruction streams and data streams
- The parallel organizations are characterized by the multiplicity of hardware provided to service I and D streams:
  - SISD: Single Instruction and Single Data Stream (uniprocessor)
  - SIMD: Single I and Multiple D Streams (GPUs)
  - MISD: Multiple I and Single D Streams
  - MIMD: Multiple I and Multiple D Streams (multicores)

# Model-Based Classification

- Shared-memory
- Message-passing
- Dataflow
- Systolic
- Data parallel

# Shared-Memory Architectures

- Key feature: all processors in the system can directly access all memory locations in the system, thus providing a convenient and cheap mechanism for multiple processors to share data
  - Convenient: (i) location transparency, (ii) abstraction supported is same as that of uniprocessors
  - Cheap: as compared to other models (more later)
- Memory can be centrally placed or distributed
- Better name is Single address space machines

- Programming model:
  - Variety of parallel models can be easily supported: fork-join model, task queue model, data parallel model.
  - Parallel threads use shared-memory for communication and synchronization
- A problem traditionally cited with such machines is scalability
- However, the programming model is very general and can easily emulate the other models

# Message Passing Architectures

- Processors can directly access only local memory, and all comm. and synch happens via messages
- Sending a message often incurs many overheads:
  - Building a header, copying data into network buffers, sending data, receiving data into buffers, copying data from kernel to user process address space
  - Many of these steps may require OS intervention
- Synchronization using messages is typically based on various handshake protocols
- One of the main advantages is easy scalability

# Message Passing Architectures

- A variety of programming models can be supported:
  - Actor model, Concurrent-object oriented programming
- Very popular:
  - Clusters
  - Cloud computing
  - Supercomputing

# Dataflow

- In the dataflow model, instructions are activated by the availability of data operands for instructions

  A = (B+1)* (B-C)

  D = C * E

  F = A * D

- In control flow models, computation is a series of instructions with implicit or explicit sequencing between them
- One of the advantages is that all dependences are explicitly present in the dataflow graph, so parallelism is not hidden from hardware

# Dataflow

- Many variations exist
- Some issues:
  - Granularity of operations (locality issues)
  - Efficient handling of complex data structures like arrays
  - Complexity of the matching store
  - Problems due to excess of parallelism

# Systolic Architectures

- Basic principle:
  - Replace a single PE by a regular array of PEs and carefully orchestrate the flow of data between Pes
  - Result: high throughput without increasing memory bandwidth requirements
- Distinguishing features from regular pipelined computers:
  - Array structure can be non-linear (e.g. hexagonal)
  - Pathways between the PEs may be multidirectional
  - PEs may have local instruction and data memory, and are more complex than the stage of a pipelined computer

# Systolic Architectures

- Issues:
  - System integration: Shipping data from host array and back
  - Cell architecture and communication architecture
  - Software for automatically mapping computations to systolic arrays
  - General purpose systolic arrays
- Popular in signal processing

# Data Parallel Architectures

- Programming model assumes that there is a processor associated with each member of a collection of data
- All processors execute similar operations on different data, not in lockstep
- Useful for highly parallel codes
- Used by graphics processors