## Multiprocessors - Flynn's Taxonomy (1966)

- Single Instruction stream, Single Data stream (SISD)
  - Conventional uniprocessor
  - Although ILP is exploited
    - Single Program Counter -> Single Instruction stream
    - The data is not "streaming"
- Single Instruction stream, Multiple Data stream (SIMD)
  - Popular for some applications like image processing
  - One can construe vector processors to be of the SIMD type.
  - MMX extensions to ISA reflect the SIMD philosophy
    - Also apparent in "multimedia" processors (Equator Map-1000)
  - "Data Parallel" Programming paradigm



## Flynn's Taxonomy (c'ed)

- Multiple Instruction stream, Single Data stream (MISD)
  - Don't know of any
- Multiple Instruction stream, Multiple Data stream (MIMD)
  - The most general
  - Covers:
    - Shared-memory multiprocessors
    - Message passing multicomputers (including networks of workstations cooperating on the same problem)



## Shared-memory Multiprocessors

- Shared-Memory = Single shared-address space (extension of uniprocessor; communication via Load/Store)
- Uniform Memory Access: UMA
  - Today, almost uniquely in shared-bus systems
  - The basis for SMP's (Symmetric MultiProcessing)
  - Cache coherence enforced by "snoopy" protocols
  - Form the basis for clusters (but in clusters access to memory of other clusters is not UMA)



## Shared-memory Multiprocessors (c'ed)

- Non-uniform memory access: NUMA
  - NUMA-CC: cache coherent (directory-based protocols or SCI)
  - NUMA without cache coherence (enforced by software)
  - COMA: Cache Memory Only Architecture
  - Clusters
- Distributed Shared Memory: DSM
  - Most often network of workstations.
  - The shared address space is the "virtual address space"
  - O.S. enforces coherence on a page per page basis



## Message-passing Systems

- Processors communicate by messages
  - Primitives are of the form "send", "receive"
  - The user (programmer) has to insert the messages
  - Message passing libraries (MPI, OpenMP etc.)
- Communication can be:
  - Synchronous: The sender must wait for an ack from the receiver (e.g, in RPC)
  - Asynchronous: The sender does not wait for a reply to continue



## Shared-memory vs. Message-passing

- An old debate that is not that much important any longer
- Many systems are built to support a mixture of both paradigms
  - "send, receive" can be supported by O.S. in shared-memory systems
  - "load/store" in virtual address space can be used in a message-passing system (the message passing library can use "small" messages to that effect, e.g. passing a pointer to a memory area in another computer)
  - Does a network of workstations with a page being the unit of coherence follow the shared-memory paradigm or the message passing paradigm?

## The Pros and Cons

- Shared-memory pros
  - Ease of programming (SPMD: Single Program Multiple Data paradigm)
  - Good for communication of small items
  - Less overhead of O.S.
  - Hardware-based cache coherence
- Message-passing pros
  - Simpler hardware (more scalable)
  - Explicit communication (both good and bad; some programming languages have primitives for that), easier for long messages
  - Use of message passing libraries



## Caveat about Parallel Processing

- Multiprocessors are used to:
  - Speedup computations
  - Solve larger problems
- Speedup
  - Time to execute on 1 processor / Time to execute on N processors
- Speedup is limited by the communication/computation ratio and synchronization
- Efficiency
  - Speedup / Number of processors

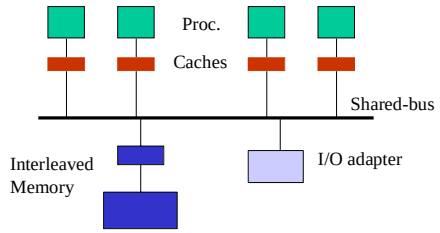


## Amdahl's Law for Parallel Processing

- Recall Amdahl's law
  - If x% of your program is sequential, speedup is bounded by 1/x
- At best linear speedup (if no sequential section)
- What about superlinear speedup?
  - Theoretically impossible
  - "Occurs" because adding a processor might mean adding more overall memory and caching (e.g., fewer page faults!)
  - Have to be careful about the x% of sequentiality. Might become lower if the data set increases.
- Speedup and Efficiency should have the number of processors and the size of the input set as parameters

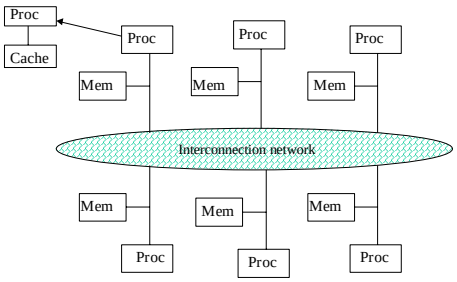


## SMP (Symmetric MultiProcessors aka Multis) Single shared-bus Systems






## Multiprocessor with Centralized Switch






## Multiprocessor with Decentralized Switches