CS546 Parallel and Distributing Processing

- Instructor: Professor Xian-He Sun
  - Email: sun@iit.edu, Phone: (312) 567-5260
  - Office hours: 4:30pm-5:30pm Tuesday, Thursday SB235C
- TA: Xiaoyang Lu
  - Email: xlu40@hawk.iit.edu
  - Office hour: 2:30pm-3:30pm MW, SB003
- Blackboard:
  - http://blackboard.iit.edu
- Additional Web site
  - http://www.cs.iit.edu/~sun/cs546.html

“Overview of Parallel Computing”

Summary

- What is parallel computing
- Why parallel computing
- Different levels of parallelism
- Challenge and opportunities of parallelism

- Reading:
  - Kumar – ch 1; ch 2

Review Question

- What is the difference between parallel processing and concurrent processing?
- Multi-tasking, multi-core

What is Parallel Processing

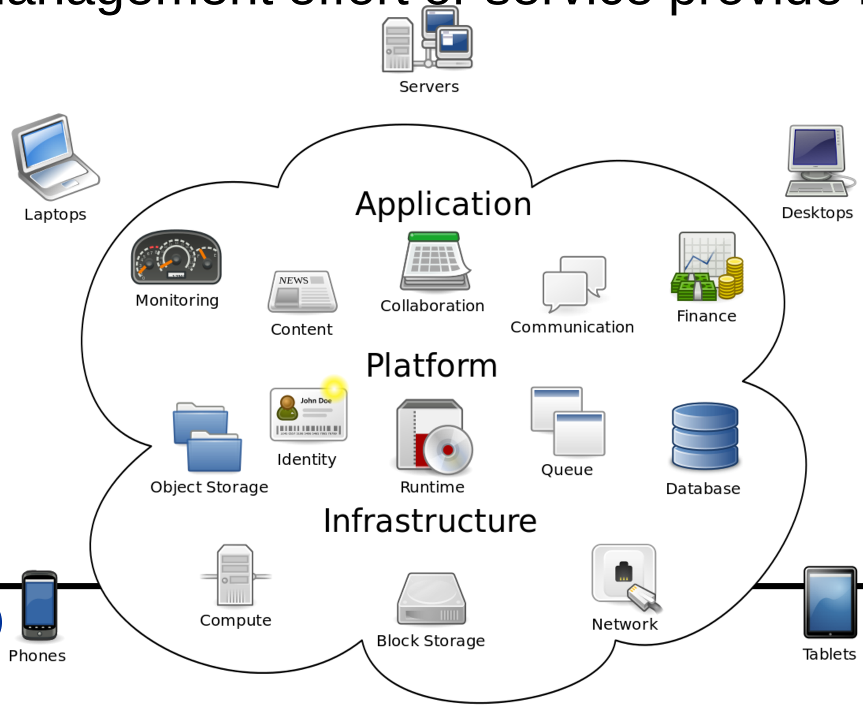
- Parallel Processing
  - Several working entities work together toward a common goal
- Parallel Processing
  - A kind of information processing that emphasizes the concurrent manipulation of data elements belonging to one or more processes solving a single problem
- Parallel Computer
  - A computer designed for parallel processing

Review Question

- What is the difference between parallel processing and concurrent processing?
  - Multi-tasking, multi-core
- What is the difference between high performance computing and Cloud computing?
- Parallel processing, distributed processing

Cloud Computing

- **Cloud computing** is Internet-based computing, whereby shared resources, software and information are provided to computers and other devices on-demand like a public utility (**Wikipedia**)
- **A cloud** is a computing capability that provides an abstraction between the computing resource and its underlying technical architecture, enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provide interaction (**NIST**)



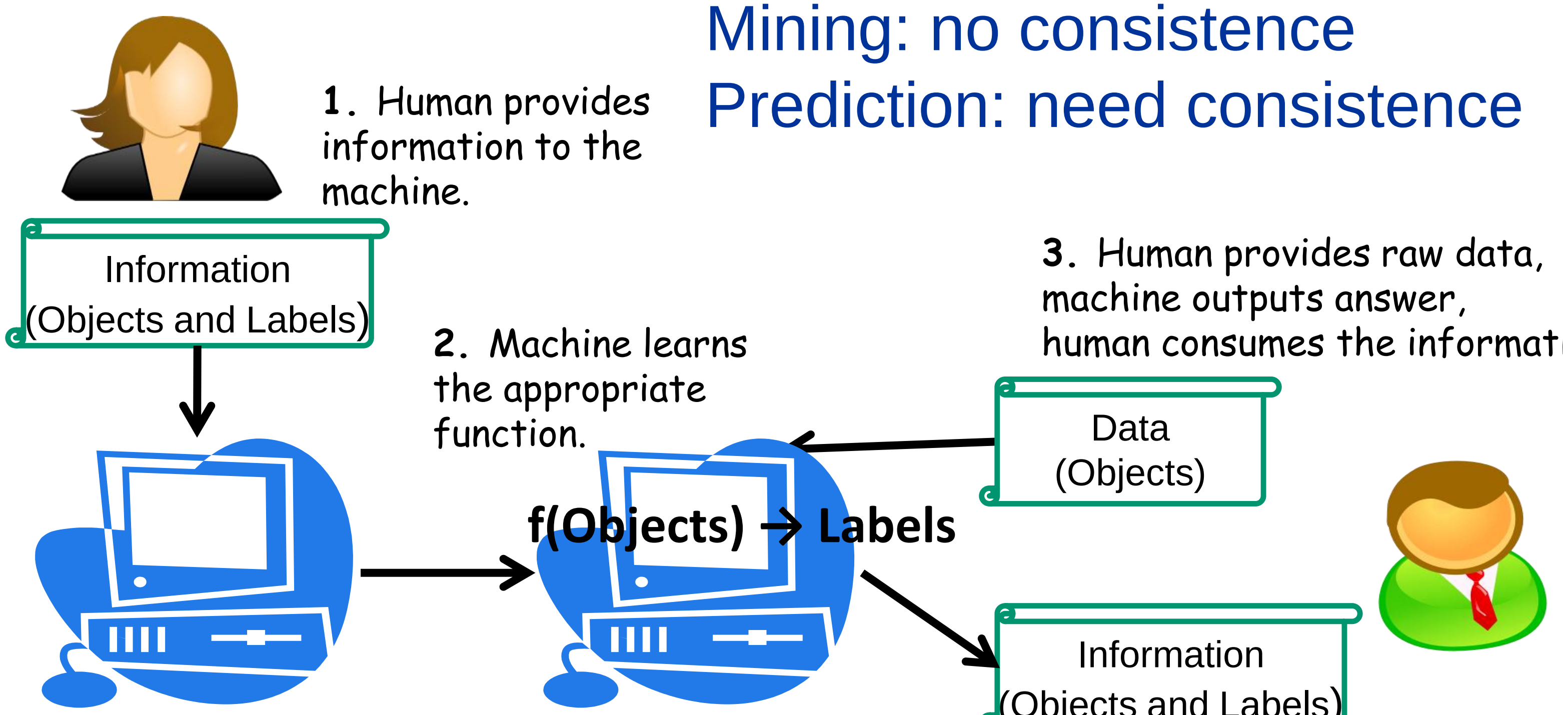
Deployment models:
- Public cloud
- Private cloud
- Hybrid cloud

Review Question

- What is the difference between parallel processing and concurrent processing?
  - Multi-tasking, multi-core
- What is the difference between high performance computing and Cloud computing?
  - Parallel processing, distributed processing
- What is the relationship between HPC and big data?
- High performance data analytic, big computing

Big Data : discover information/knowledge from data

Mining: no consistence
Prediction: need consistence



1. Human provides information to the machine.

2. Machine learns the appropriate function.

3. Human provides raw data, machine outputs answer, human consumes the information

Machine passively consumes information

Human passively consumes information

The Surge of Cloud & Big Data

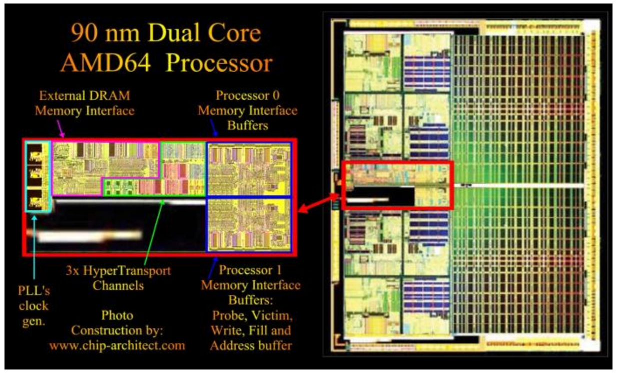
Mimic the electrical power grid

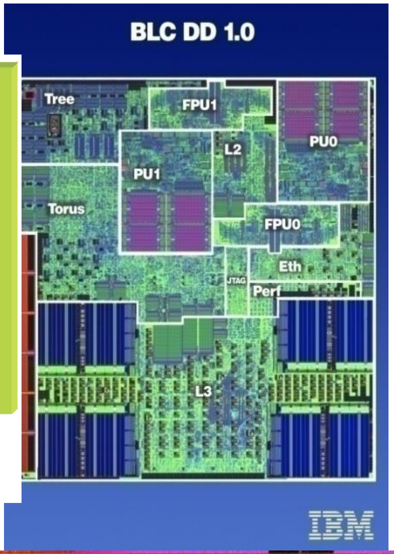
Review Question

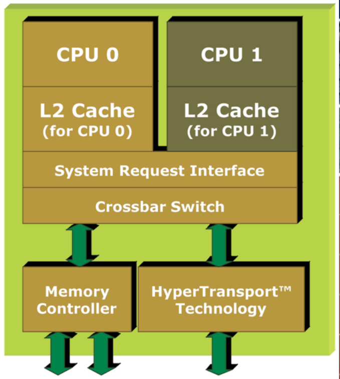
- What is the difference between parallel processing and concurrent processing?
  - Multi-tasking, multi-core
- What is the difference between high performance computing and Cloud computing?
  - Parallel processing, distributed processing
- What is the relation between HPC and big data?
  - High performance data analytic, big computing
- What are the motivations behind multi-core, many-core architectures?
- Different levels of parallelism

Multi-Core

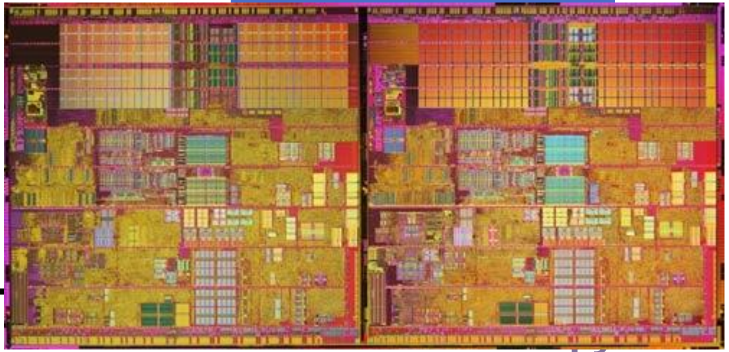
- Motivation for Multi-Core
  - Exploits improved feature-size and density
  - Increases functional units per chip (spatial efficiency)
  - Limits energy consumption per operation
  - Constrains growth in processor complexity
- Challenges resulting from multi-core
  - Aggravates memory wall
    - Memory bandwidth
      - Way to get data out of memory banks
      - Way to get data into multi-core processor array
    - Memory latency
    - Fragments L3 cache
  - Relies on effective exploitation of multiple-thread parallelism
    - Need for parallel computing model and parallel programming model
  - Pins become strangle point
    - Rate of pin growth projected to slow and flatten
    - Rate of bandwidth per pin (pair) projected to grow slowly
  - Requires mechanisms for efficient inter-processor coordination
    - Synchronization
    - Mutual exclusion
    - Context switching


90 nm Dual Core AMD64 Processor
External DRAM Memory Interface
Processor 0 Memory Interface Buffers
3x HyperTransport Channels
PLL's clock gen.
Processor 1 Memory Interface Buffers: Probe, Victim, Write, Fill and Address buffer
Photo Construction by: www.chip-architect.com





AMD Athlon™ 64 X2 Dual-Core Processor Design



Why Advanced (Parallel) Computing

- Application driven
  - New demand of existing applications
  - New applications with different demand
- Technology driven
  - New technologies come out
  - Fundamental limits are being approached of existing technology

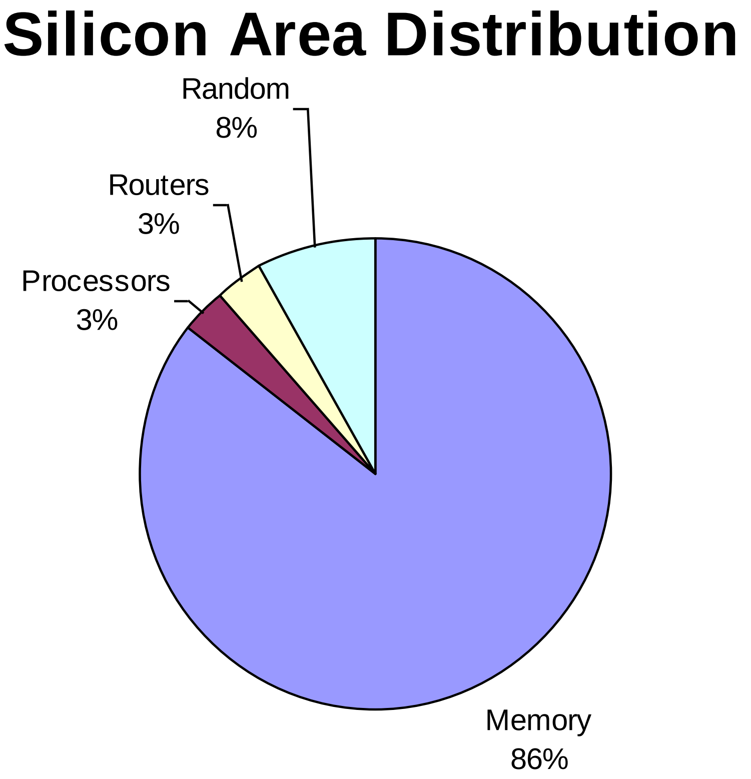
An exciting and dynamic field

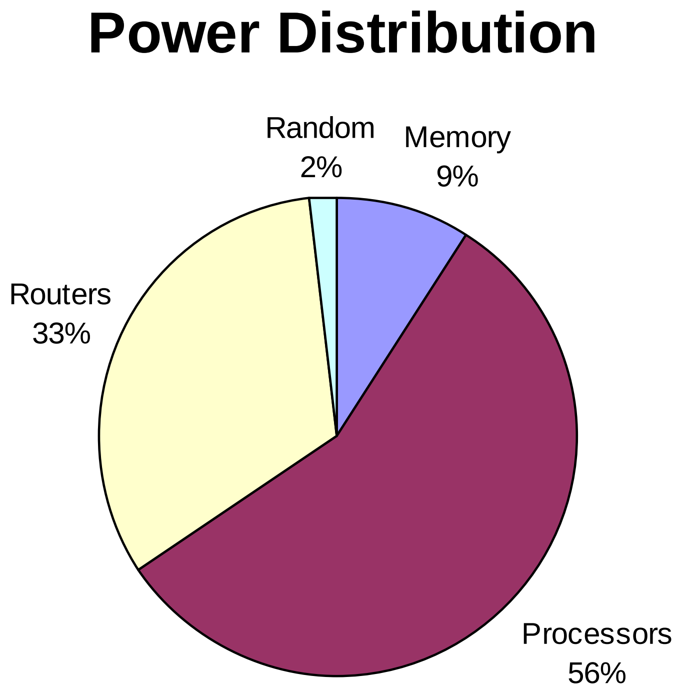
Outline

- Overview of parallel architectures
  - SISD, SIMD, MISD, MIMD
- Architectures:
  - Shared mem. Vs. distributed mem.
- Architectures:
  - Interconnects
- Software parts: OSs, compilers,…
- Flavors of parallelism
- Challenges

- Homework: Reading Kumar – Chapter 1 & 2

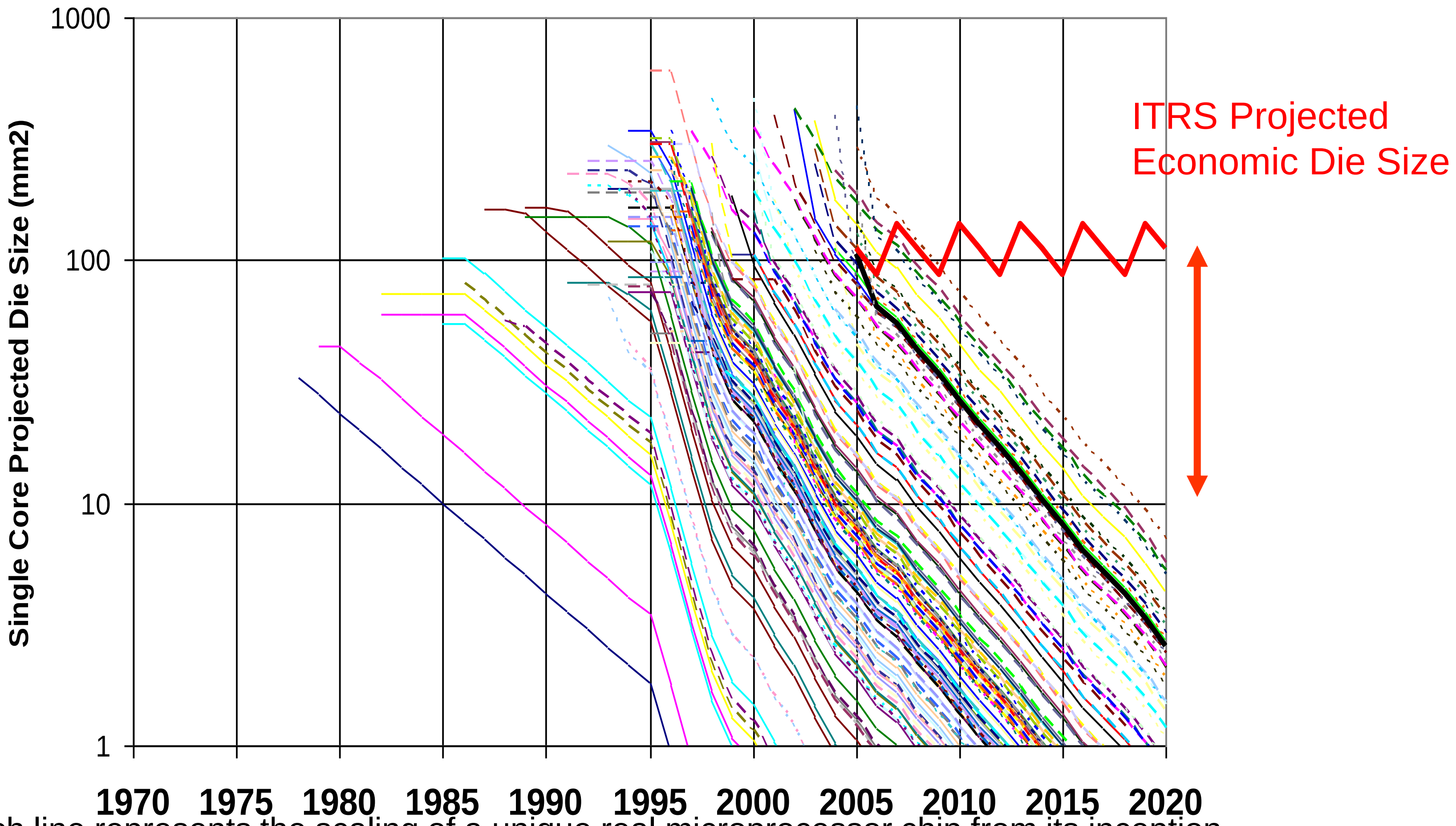
What Are We Doing with the Total System Silicon?





Courtesy of Peter Kogge, UND

Area Scaling Alone Reveals the Rationale for Multi-Core

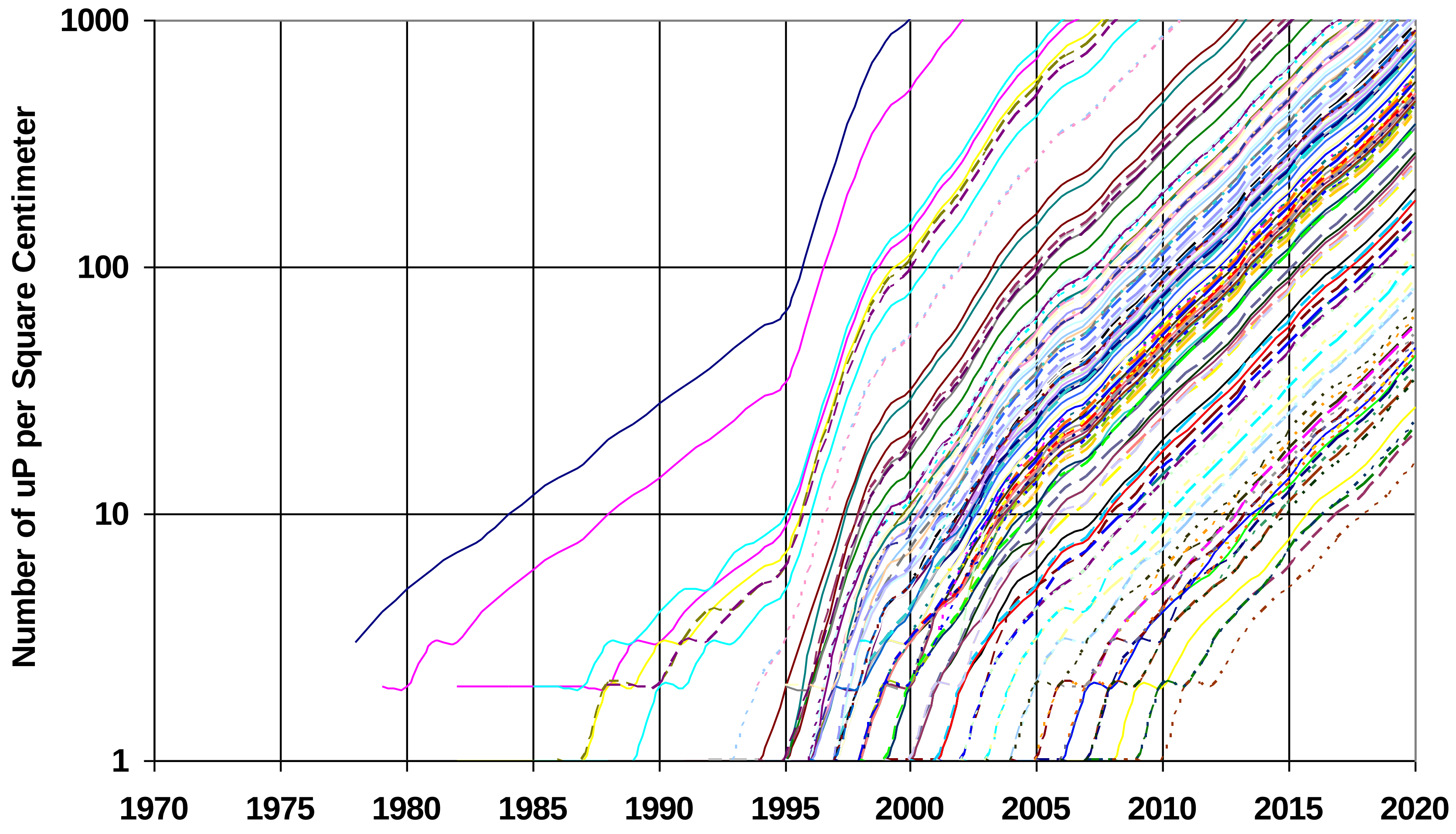


Each line represents the scaling of a unique real microprocessor chip from its inception

Courtesy of Peter Kogge, UND

How Many Can We Fit on a cm^2?

Assume we scale entire current single core chip & replicate to fill 280 sq mm die

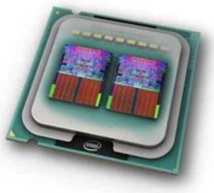


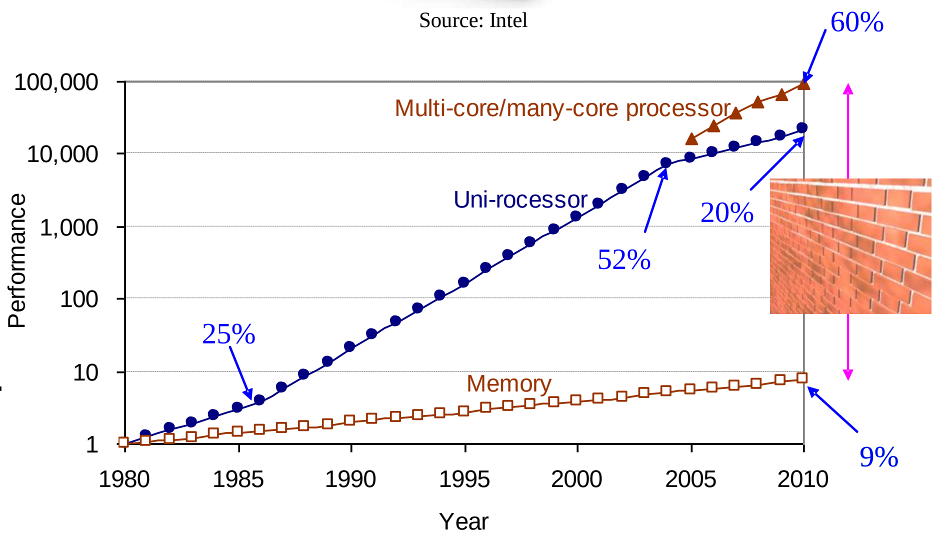
Answer Potentially 1000's!!!!

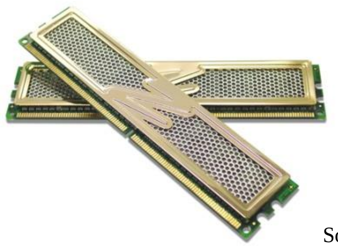
Courtesy of Peter Kogge, UND

Problem: *The Memory-wall Problem*

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004
  - Aggregate multi-core/many-core processor performance even higher since 2004
- Memory: ~9% per year
- I/O: ~6% per year
- Processor-memory speed gap keeps increasing

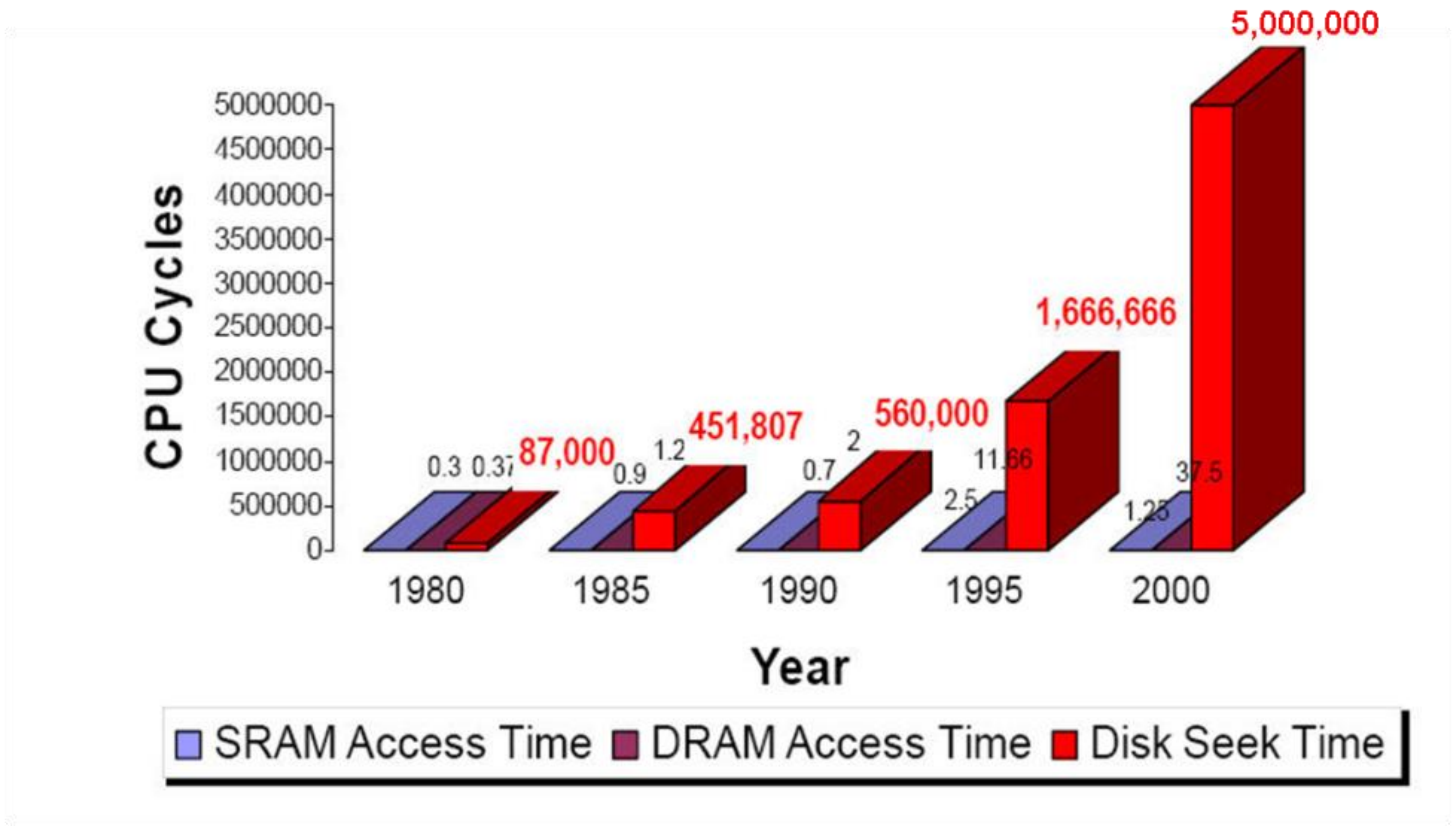

Source: Intel




Source: OCZ

Memory-bounded speedup (1990), Memory wall problem (1994)

I/O Bottleneck



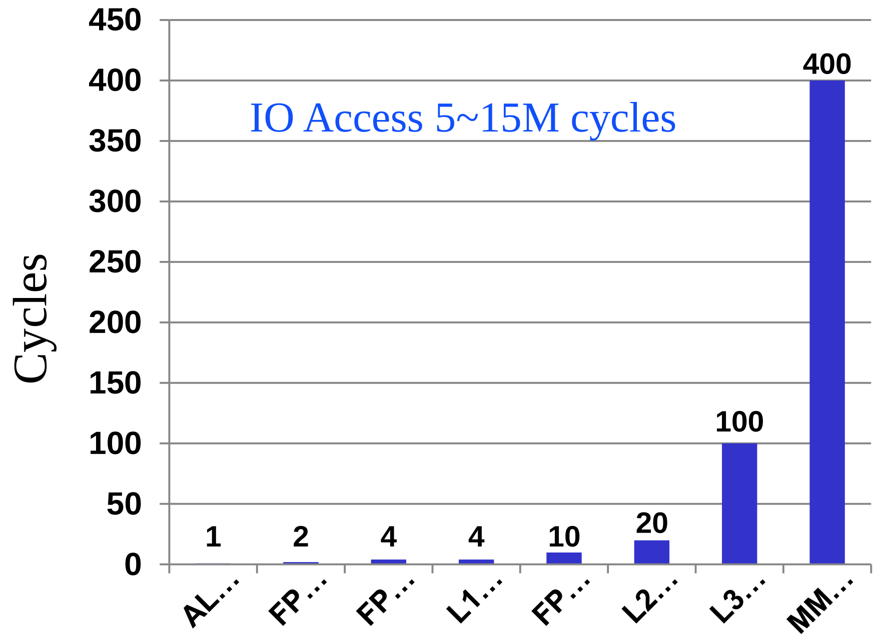
Bryant and O'Hallaron, "*Computer Systems: A Programmer's Perspective*", Prentice-Hall 2003

Assumption of Current Solutions

- ❑ Memory Hierarchy: Locality
- ❑ Concurrence: Data access pattern
  - ○ Data stream

Extremely Unbalanced Operation Latency

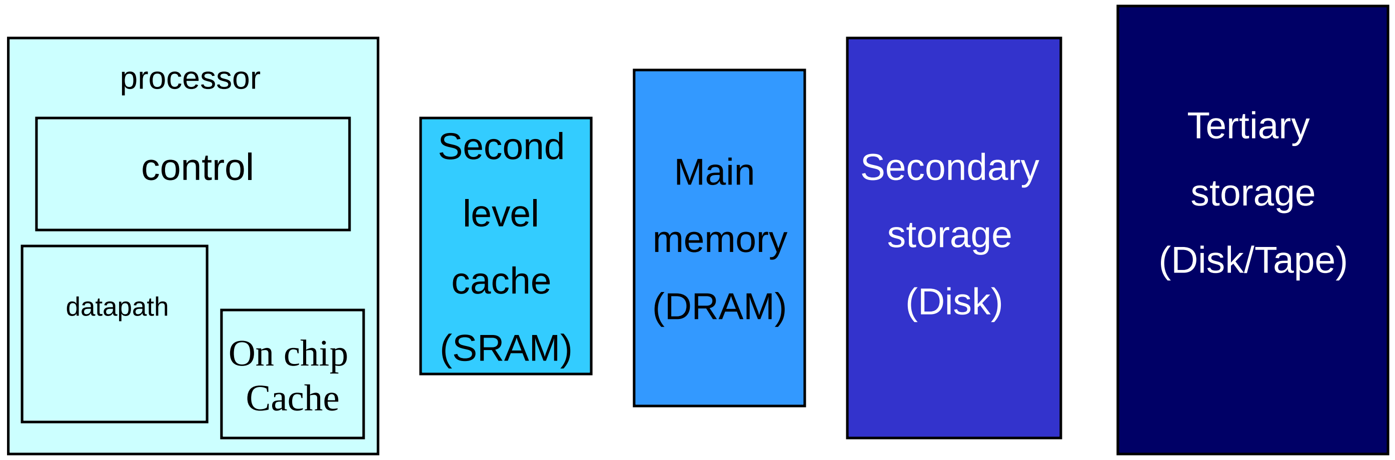
Performances vary largely



Levels of the Memory Hierarchy

Deeper levels of cache memory

Large memories are slow, fast memories are small and expensive.



Speed (ns):	1-5 ns	5 – 25 ns	80 - 250ns	~10 ms	~ 10 sec
Size (bytes):	32KB	1-2MB	1-8GB	20-200GB	TB

The Principle of Locality

- The Principle of Locality:
  - Programs access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon
    (e.g., straight line code, array access)
    - Cache Block or Cache Line
- Last 25 years, HW relied on locality for speed

Aspects of Parallel Computing

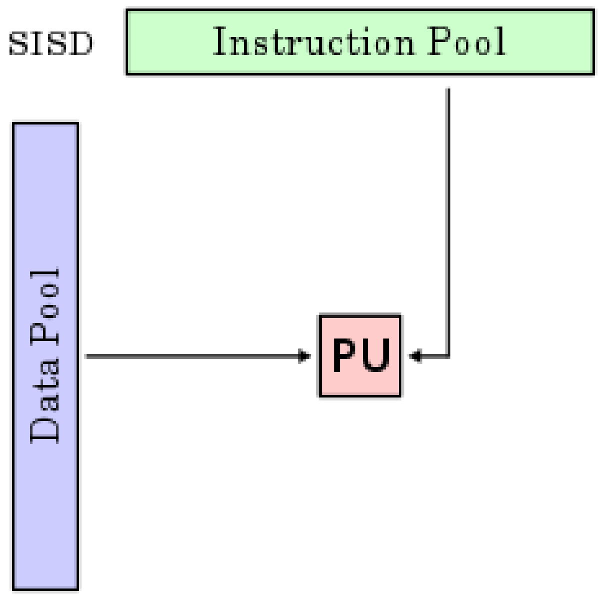
- Architectures:
  - Processors and memories connected together
- Software:
  - Operating systems
  - Compiler
  - Libraries
  - Tools – debuggers,performance analysis
- Algorithms:
  - Designing software that best fits underlying architecture

Architecture

- Basic components of any architecture:
  - Processors and memory (processing units)
  - Interconnect
- Logic classification based on:
  - Control mechanism (Flynn's Taxonomy)
    - SISD (Single Instruction Single Datastream)
    - SIMD (Single Instruction Multiple Datastream)
    - MISD (Multiple Instruction Single Datastream)
    - MIMD (Multiple Instruction Multiple Datastream)
  - Address space organization
    - Shared Address Space
    - Distributed Address Space

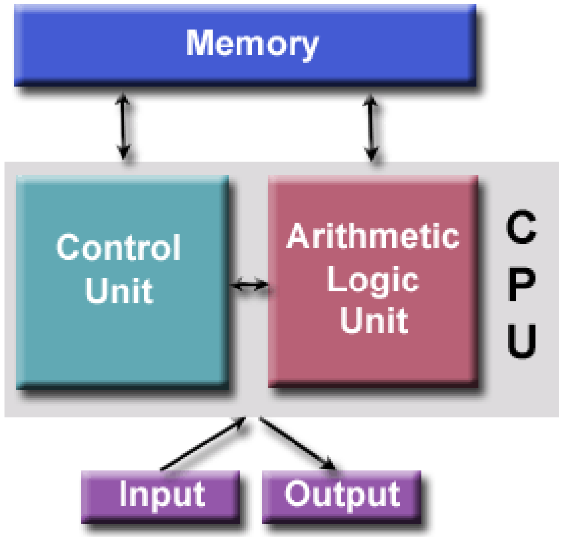
SISD Architecture

- Model of serial Von Neumann machine
- Logically, single control processor
- Includes some supercomputers, such as the 1963 CDC6600 (perhaps the first supercomputer)



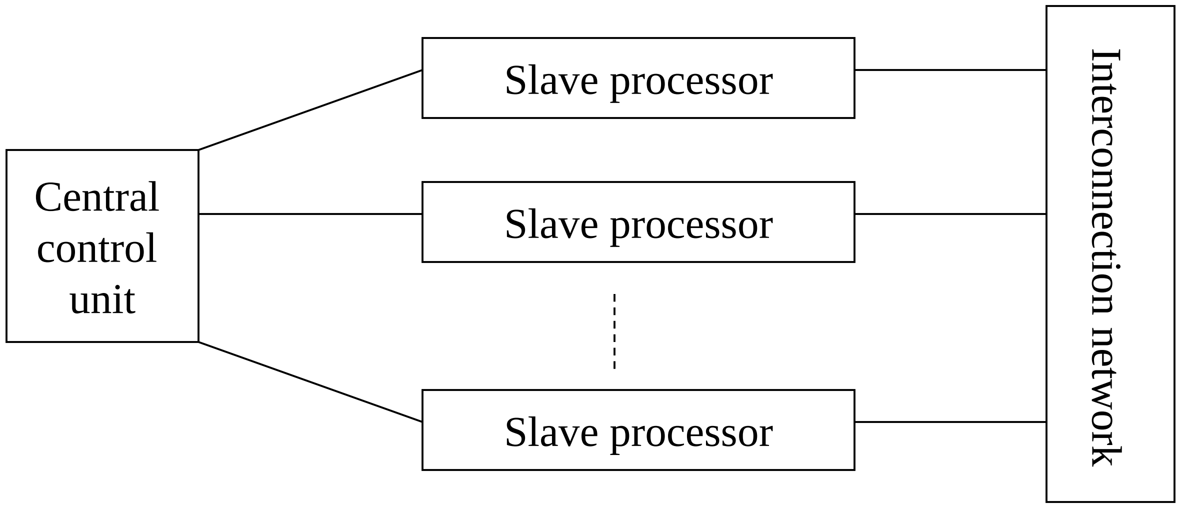
Von Neumann Architecture

- John von Neumann first authored the general requirements for an electronic computer in 1945
- Aka “stored-program computer”
  - Both program inst. and data are kept in electronic memory
- Since then, all computers have followed this basic design
- Four main components: memory, control unit, ALU, I/O



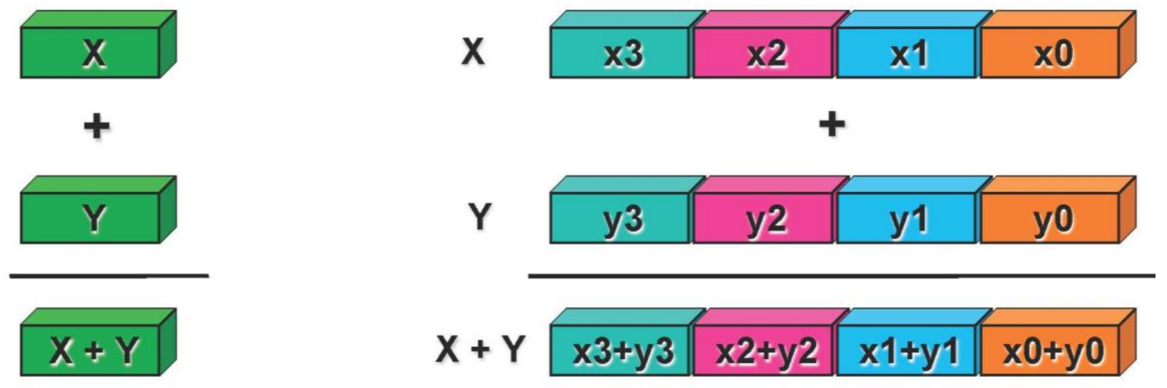
SIMD Architecture

- Multiple processors execute the same program in lockstep
- Data that each processor sees may be different
- Individual processors can be turned on/off at each cycle (“masking”)
- Examples: Illiac IV, Thinking Machines’CM-2, DAP, …



Example of SIMD Vector Units

- Scalar processing
  - Traditional mode
  - One operation produces one result
- SIMD vector units
  - One operation produces multiple results

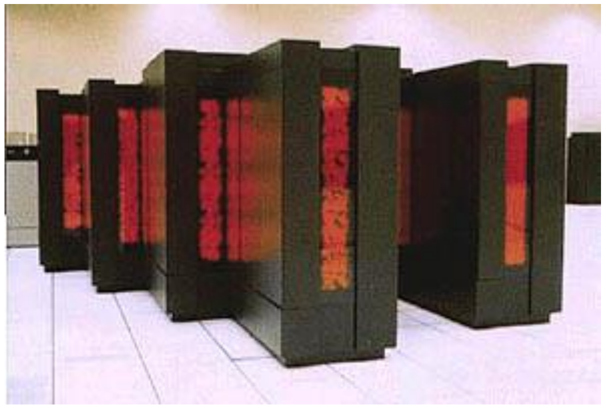


SIMD Drawbacks

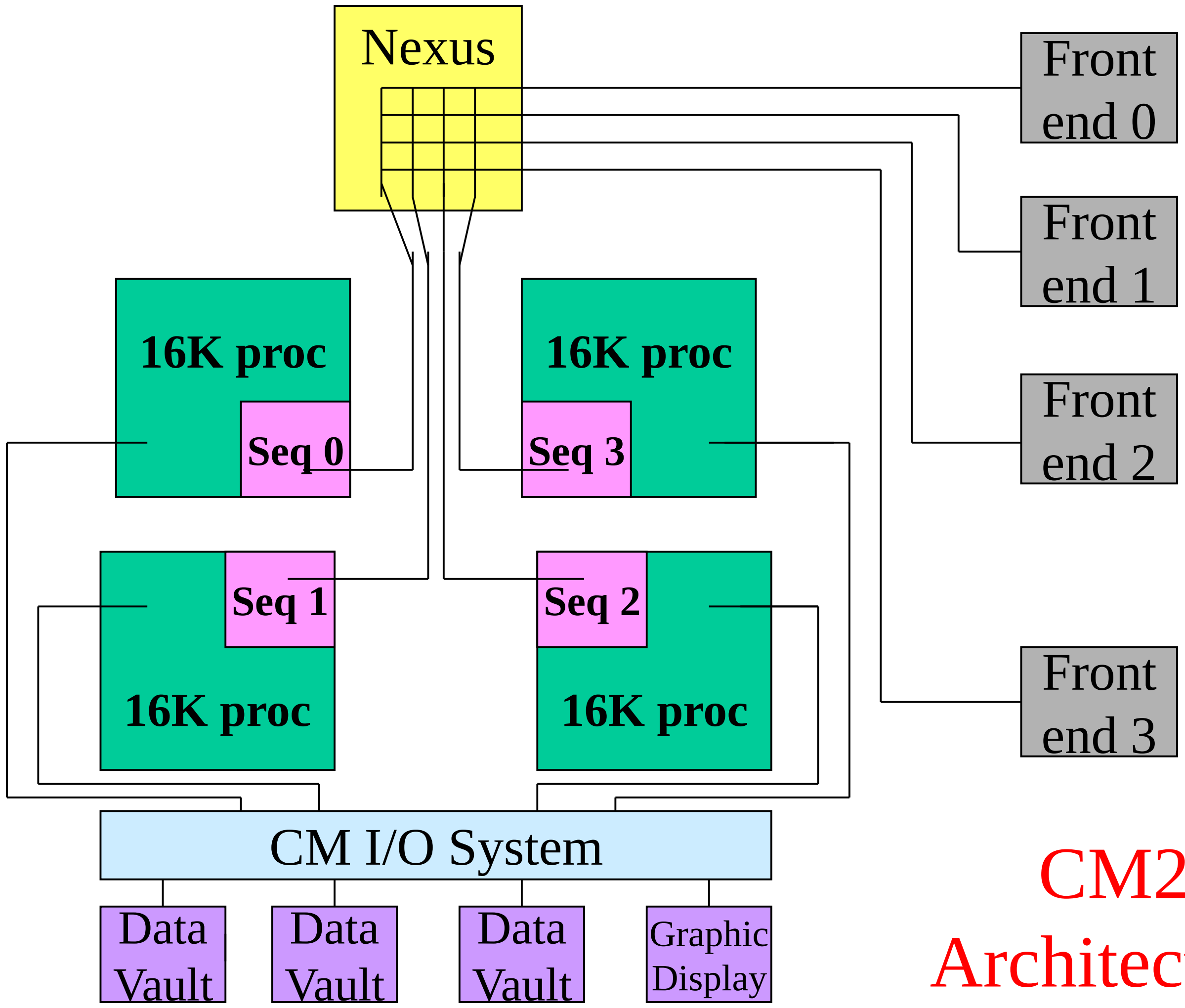
- Discussion?

Thinking Machine CM2

- CM2 (1990, built by Thinking Machines Corp) had 8,192 to 65,536 one-bit processors, plus one floating-point unit.
- Data Vault provides peripheral mass storage
- Single program - all unmasked operations happened in parallel.



Thinking Machine CM2



CM2 Architecture

Vector Processors

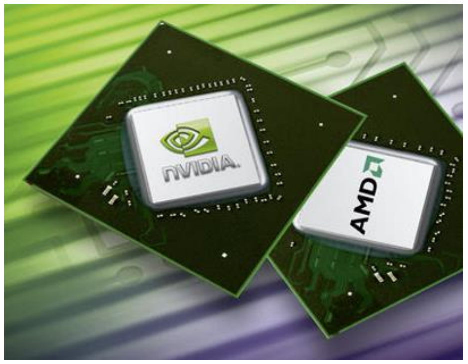
- Operate on arrays or vectors of data, while conventional CPU's operate on individual data elements or scalars
- Vector registers
  - Capable of storing a vector of operands and operating simultaneously on their contents
- Vectorized and pipelined functional units
  - The same operation is applied to each element in the vector
- Examples:
  - Cray supercomputers (X-MP, Y-MP, C90, T90, SV1, ...), Fujitsu (VPPxxx), NEC, Hitachi
  - Earth Simulator from Japan (on the TOP500 list)
  - many of these have multiple vector processors, but typically separate processors are used for separate jobs.

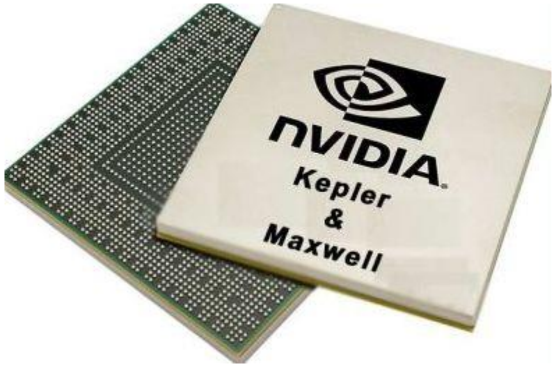
Vector Processors – Pros & Cons

- Discussion?

Graphic Processing Units (GPU)

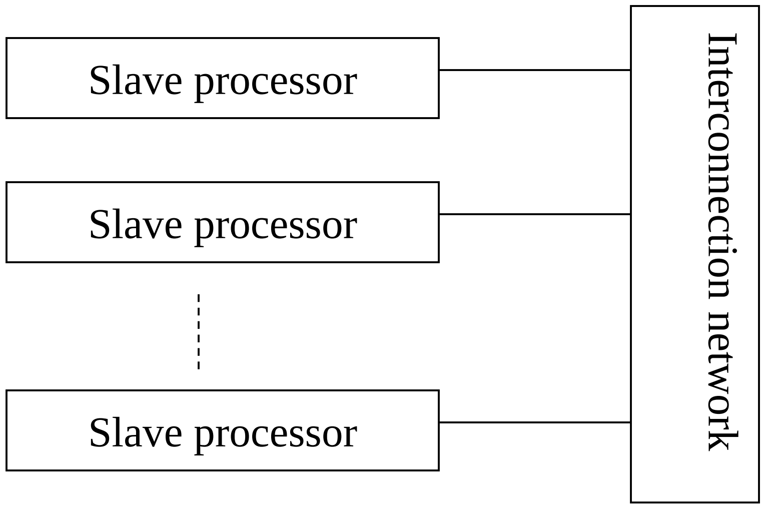
- Real time graphics application programming interfaces or API's use points, lines, and triangles to internally represent the surface of an object
- A graphics processing pipeline converts the internal representation into an array of pixels that can be sent to a computer screen
- Several stages of this pipeline (called shader functions) are programmable
  - Typically just a few lines of C code





MIMD Architecture

- Each processor executes program independent of other processors
- Processors operate on separate data streams
- May have separate clocks
- Examples: IBM SP, TMC's CM-5, Cray T3D & T3E, SGI Origin, Tera MTA, …
- SPMD (Single Program Multiple Data)



MISD Architectures

- Multiple Instruction Single Data
- The term isn’t used (except when discussing the Flynn taxonomy) .
- Perhaps applies to pipelined computation, e.g. sonar data passing through sequence of special-purpose signal processors.

SIMD vs. MIMD

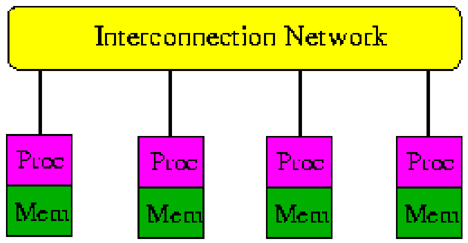
- SIMD:
  - Need custom hardware for processors
  - Slave processors are simple and therefore low cost
  - Good for programs with lots of synchronization due to lock step operation
- MIMD:
  - Easy to build out of commodity parts
  - Processors are complex, but availability of low cost commodity microprocessors offsets the SIMD advantage
  - Can handle a more general class of problems with reasonable efficiency

Parallel Architectures

IBM RS/6000 SP

SGI Power Challenge XL

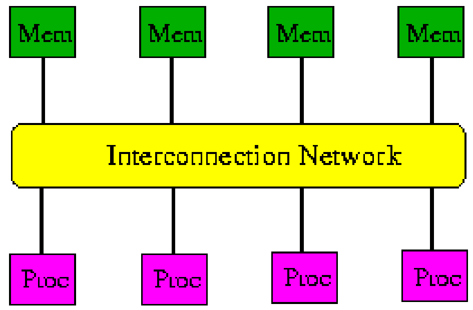
Distributed Memory Machines



Advantages:
+ scalable
+ latency hiding

Disadvantages:
- harder to program
- program must be replicated

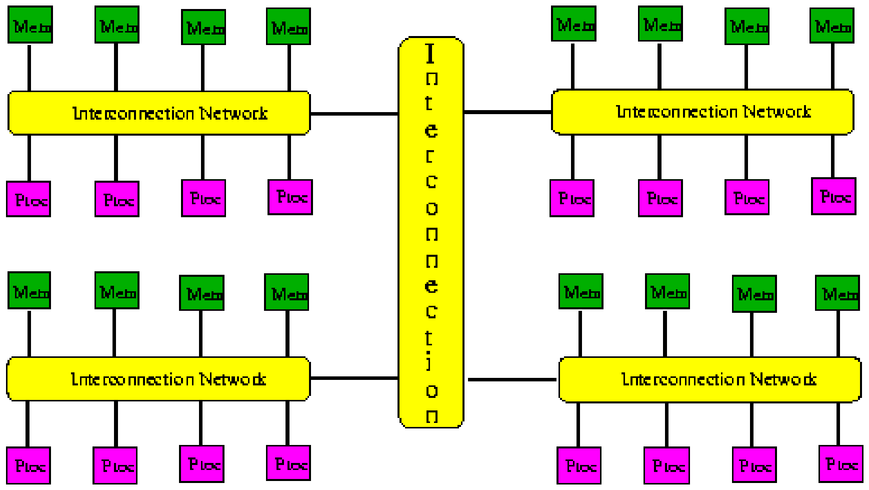
Shared Memory Machines



Advantages:
+ ease of programming
+ processors share code and data

Disadvantages:
- scalability problem

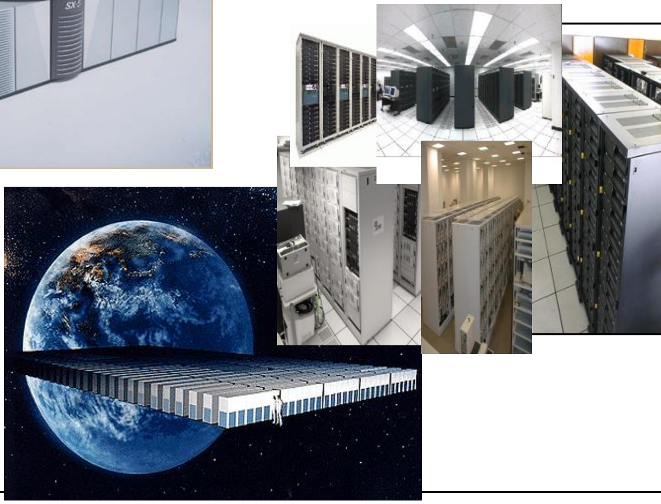
Cluster of Symmetric Multiprocessor Systems (SMP)



Advantages:
+ scalable

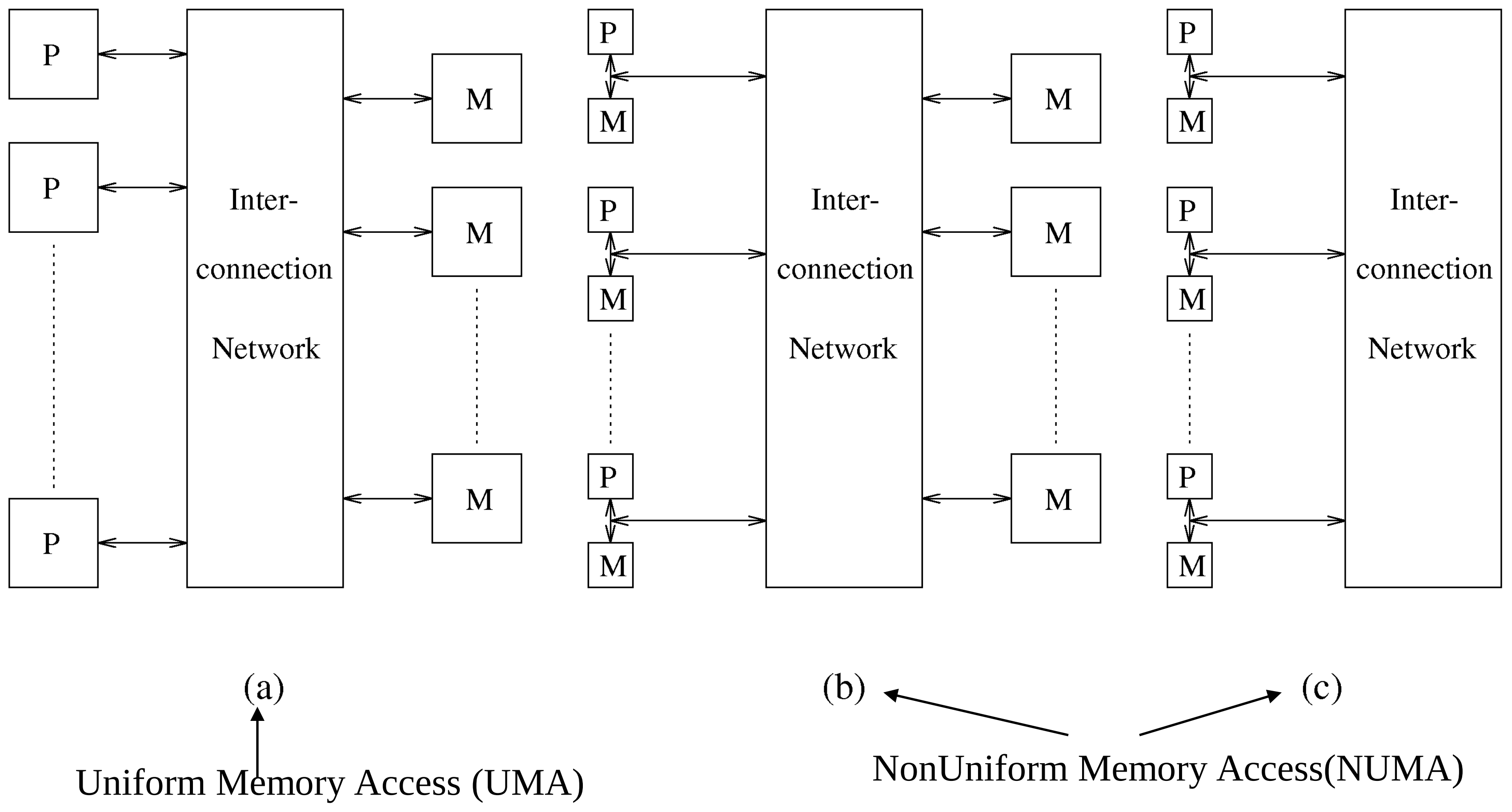
Disadvantages:
- programming paradigm unclear

NEC SX-5 multi node
(8 CPUs pro Knoten)



Earth Simulator
(540*8 CPUs)

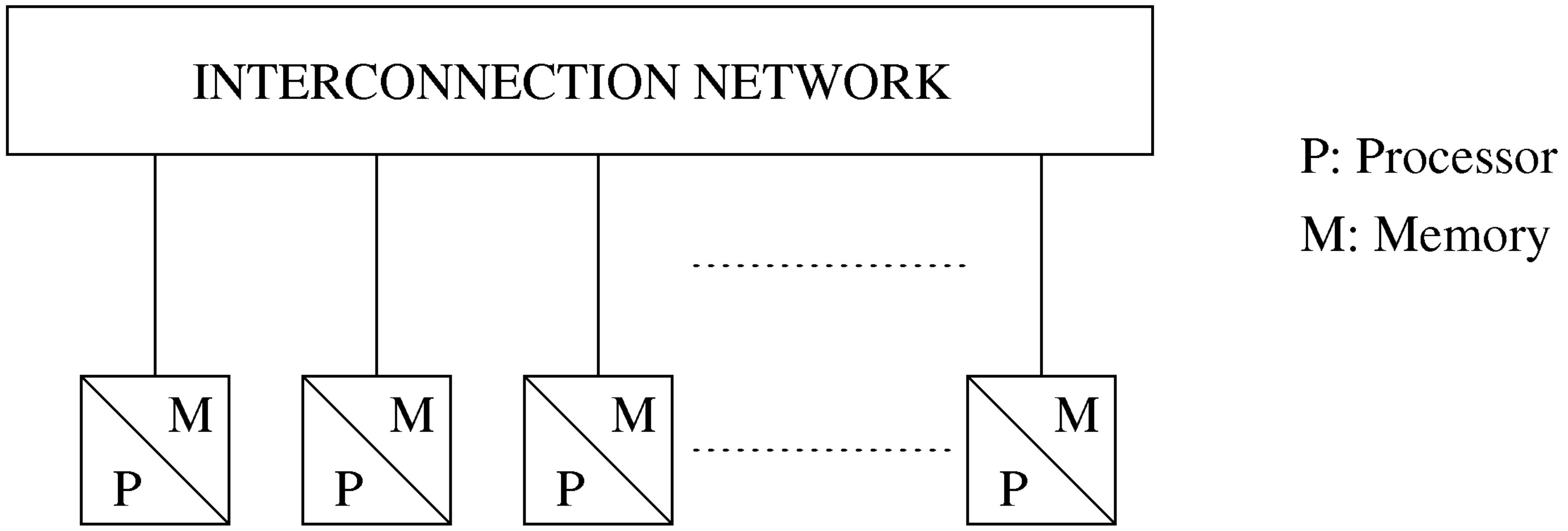
Shared Memory Architecture



NUMA (non-uniform memory access)

- Every processor has memory and cache, together form a global address space, where local access is much faster than remote access
- Cache-coherent NUMA - ccNUMA: Home location of data is fixed. Copies of shared data in the processor caches are automatically kept coherent, i.e. new values are automatically propagated. (SGI Origin/Altix 3000/ASCI Blue Mountain, Convex SPP)
- Non-cache-coherent NUMA - nccNUMA: Home location of data is fixed. Copies of shared data are independent of original location. (Cray T3E)
- Cache-only memory - COMA: Data migrate between memories of the nodes, i.e. home location can be changed. (KSR-1 computer)

Distributed Memory Architecture

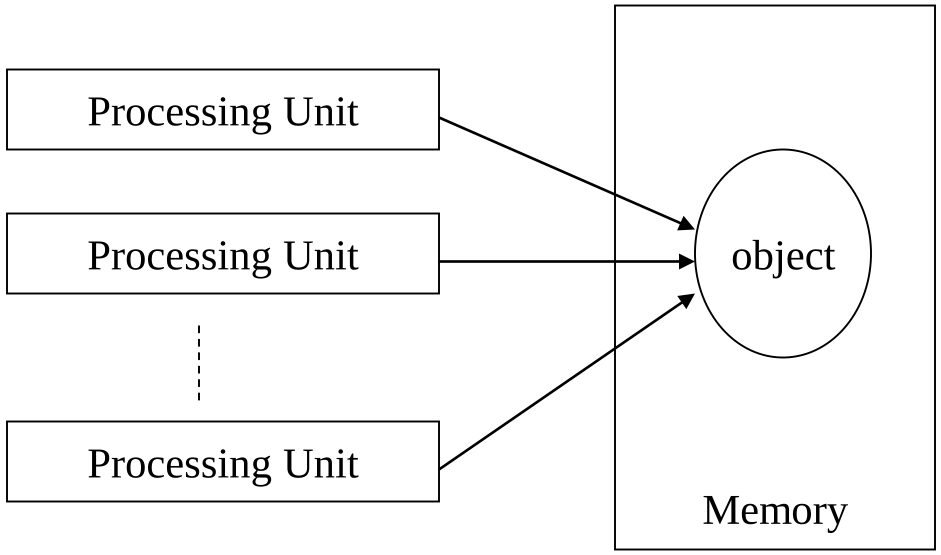


MIMD computers

- MIMD computers
  - Distributed Memory - DM (multicomputer): Building blocks are nodes with private physical address space. Communication is based on messages.
  - Shared Memory - SM (multiprocessor): System provides a shared address space. Communication is based on read/write operation to global addresses.
    - Symmetric multiprocessors - SMP (Uniform Memory Access - UMA): centralized shared memory, accesses to global memory from all processors have same latency.
    - Non-uniform Memory Access Systems - NUMA (Distributed Shared Memory Systems - DSM): memory is distributed among the nodes, local accesses much faster than remote accesses.

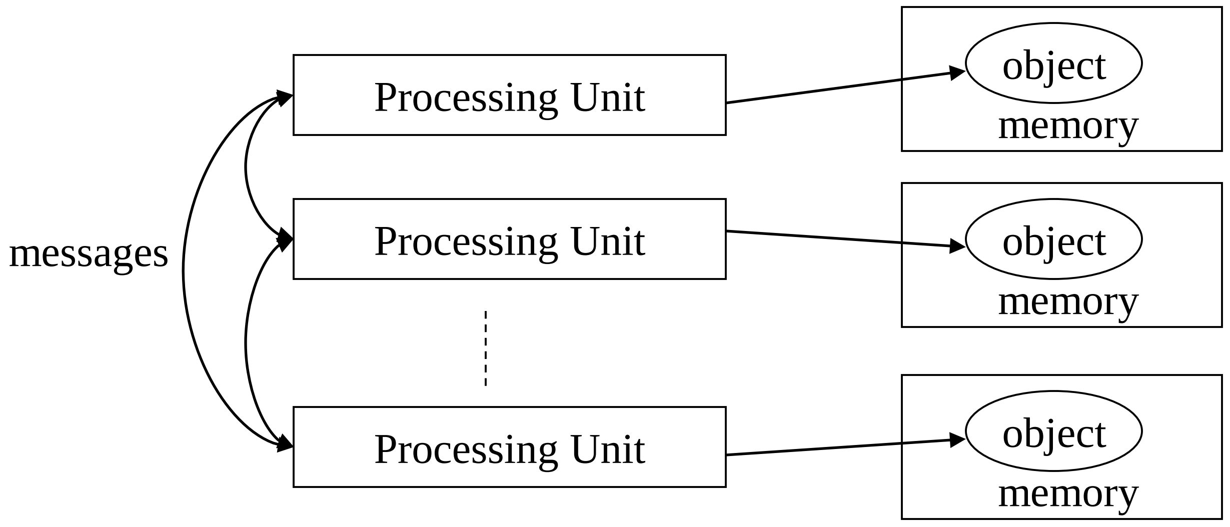
Shared Address Space

- Shared address space:
  - Processors can directly access all the data in the system
  - Inter-processor Interaction ?
  - Multi-core processor, on-chip multiprocessor



Private Address Space

- Distributed address space:
  - "Shared nothing:" each processor has a private memory
  - Processors can directly access only local data
  - Interaction ?



Shared vs. Private

- Shared memory naturally fit Shared address:
  - Pro. Vs Con.?
  - In architecture and in memory address?
- Distributed memory naturally fit Private address:
  - Pro. Vs. Con.?
  - In architecture and in memory address?
- Hybrid model
  - Logically shared physically distributed

Shared Address Space MIMD

- Typically *time shared*
- Access to job queue can be centralized or decentralized
- Parallel programming support ?
- Data access delay?

Private Address Space MIMD

- Usually access to parallel machine is via a host computer running a serial OS
- Nodes of parallel machine have a simple version of OS
- Typically *space shared*
- Parallel programming support ?
- Communication delay?

- Interconnection (switch): A primary component of parallel computers

Disjoint Private Address Space

ADVANTAGES

- Simple to develop
- Performance (local access)
- easily/ readily expandable
- highly reliable (any CPU failure does not affect the whole system)

DISADVANTAGES

- Difficulty of programming
- Performance (remote access)

Variants

- Combination of both concepts
  - Disjoint address space (local memory) + Global address space
  - Two level architectures: subsets of processors having a global shared space limited to that subset!!

- SMP clusters (Uniform Memory Access - UMA):
  - centralized shared memory, accesses to global memory from all processors have same latency.

- Shared Virtual Memory: at the programmer's level, the memory space is shared but at the physical level, address spaces are disjoint.

  e.g. KSR-1