Distributed [*Computing*] Systems

Introduction

Isaac D. Scherson (aka The Schark)

Dept. of Computer Science (Systems)
Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425

isaac@ics.uci.edu
www.ics.uci.edu/~isaac www.ics.uci.edu/~schark

CompSci-230, Winter 2019

Introduction to the course

History and Evolution

What is a Computer?

- Find the zeroes in $\mathbb{R}$ of the following polynomial:

$$2x^7 - 5x^6 - x^5 + 3x^4 - 7x^3 + x^2 + 7$$

In other words, the values of $x \in \mathbb{R}$ for which the value of the polynomial is zero.

The answer is in your computer !!!

What is a Computer?

- Can you provide a definition?
  - Let's discuss.

What is a Computer?

- A computer is a tool invented to carry out the solution to problems that would be too time-consuming and/or too laborious to solve by hand.
  - But, you must know how to solve a problem if you want to program a computer to do the solution for you !!
  - Recall that computers are only capable of basic binary arithmetic and logic (in fact, addition of two's complement numbers and AND/OR/Invert).

What is a High Performance Computer?

- The most powerful computing system feasible with current technology to solve complex (computationally intensive) problems.
  - Could be a single processor system, a multi-core system, a distributed system ...

From Tanenbaum and van Steen, "A Distributed System is a collection of independent computers that appears to its users as a single coherent system."

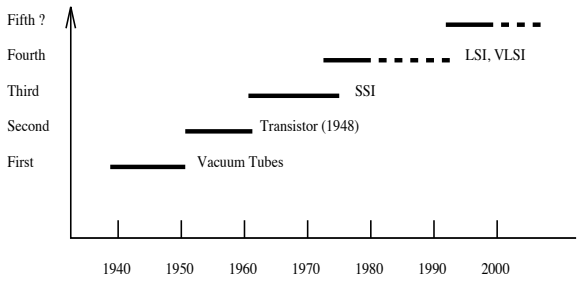
Z3 Computer (1938-1941)

- Konrad Zuse, a brilliant engineer and computer pioneer, was born in Berlin, Germany in 1910. He received his construction engineering degree from the Technische Hochschule Berlin-Charlottenburg in 1935.
- Dr. Zuse's Z3 COMPUTER, designed and built from 1938 to 1941, was the first automatic, program-controlled, fully functional, general purpose digital computer. The original Z3 was destroyed during the war. A reconstruction of the machine was made in the 1960's.
- The Z3 used binary numbers and floating point arithmetic. The Z3 also utilized a punched film for program input. The Z3 computer used 2,600 telephone relays. The Z3 could convert decimal to binary and back again.
- Dr. Konrad Zuse's pioneering work in the development of the computer was not widely known until 1965 when descriptions of his work were translated into English. His first computers pre-dated those built by Howard Aiken, John V. Atanasoff, as well as the ENIAC, built by J. Presper Eckert and John Mauchly. Zuse was unable to obtain government funding for his computer research, however, and the war effectively blocked communications between his work and that being done in other parts of the world.
- His first computers were originally called V1, V2, and V3 ("V" for "Versuchsmodell" German for experimental model). Later he changed the "V" to a "Z" so as not to be confused with Germany's V rockets.

Harvard Machine

- The Harvard architecture is a computer architecture with physically separate storage and signal pathways for instructions and data.
- The term originated from the Harvard Mark I relay-based computer, which stored instructions on punched tape (24 bits wide) and data in electro-mechanical counters.
- These early machines had data storage entirely contained within the central processing unit, and provided no access to the instruction storage as data.
- Programs needed to be loaded by an operator; the processor could not initialize itself.

Evolution of "High Performance" Computers



Generations

1st (1938-1953)

- Electronic Numerical Integrator & Computer (ENIAC)
- Electronic Discret Variable Automatic Computer (EDVAC) 1st stored program
- Assambly Language, single user, fixed point aritmetic, CPU assisted I/O

2nd (1955-1964)

- TRADIC (Bell Labs), Stretch (IBM7030, inst. lookahead & error correction), IBM 7090, CDC 1604, Univac LARC.
- HLL; FORTRAN (1956), Cobol (1959), Algol (1960), Compilers, Libraries, Batch, Monitor, Floating Point, I/O Processors, Multiplexed Memory Access

3rd (1965-1974)

- IBM 360/370, CDC 6600, TI ASC, PDP-8, ILLIAC 4 (1968) (8x8 Mesh Connected Parallel Computer)
- Microprogramming, Cache, Multiprogramming, Times-shared OS, Intelligent Compilers, Virtual Memory

4th (1975-1990)

- VAX 9000, Cray X-MP, IBM 3090, BBN TC2000, MPP (Goodyear/NASA)
- HLL for Scalar & Vector Data, Vectorizing Compilers, Languages and Environments for Parallel Processing

After K. Hwang, *Advanced Computer Architecture: Parallelism,Scalability, Programmability,* McGraw-Hill, Inc., Reading, 1993.

From the HPCI to the HPCCI and the Internet

What about the Fifth Generation?

- Japan's pre-emptive bid for the fifth generation was Artificial Intelligence with Parallel Computers.
  - LISP → Concurrent PROLOG
- USA calls for Tera-Flop Computing by mid 2000's, the fifth generation was triggered by the

HIGH PERFORMANCE COMPUTING
AND COMMUNICATIONS INITIATIVE
(HPCCI)

High Performance Computing Initiative (HPCI)

Use applications to justify/demonstrate sustained Tera-Flop performance.

- Actual problems that require numerical computer solutions and whose time complexity is too long for current super-computers
- Numerous scientific and engineering applications
- Modeling, simulation, and analysis of complex systems: e.g. climate, galaxies, molecular structures, nuclear explosions, etc.
- Business and Internet applications (although Internet did not exist yet)
  - E-commerce – ex. Amazon
  - Web servers – ex. Yahoo, Google
- Many more...

From HPCI to HPCCI and the Internet

- In June 1989, Workshop at NASA Goddard Space Flight Center to define the Grand Challenges: problems of high computational complexity whose solutions would demonstrate the feasibility of teraflop computers within 5 years. Started as the HPCI.
- A US senator representative asked that a C be added to HPCI - C for Communications.
- The senator behind the representative was Sen. Al Gore. His quest was the Information Superhighway
- The rest is Internet history !!!

The HPCCI was funded in December 1989 ... funding was distributed through (D)ARPA, NSF, DOE, NASA, HHS/NIH, DOC/NOAA, EPA, DOC/NIST.

And the technology of choice was: Scalable Parallel Computers

Grand Challenge Problems

"fundamental problems in science and engineering that have broad economic and/or scientific impact and whose solution can be advanced by applying high performance computing techniques and resources"

Originally posed by the High Performance Computing and Communications program of US Govt., many problems added by committees/agencies since then.

- Aerospace
- Computer Science
- Energy
- Environmental Monitoring and Prediction
- Molecular Biology and Biomedical Imaging
- Product Design and Process Optimization
- Space Science

The Human Genome – A great success !!!

Other examples

- Ground water remediation.
- Simulation of X-ray clusters – study of galaxy formation.
- Design and simulation of aerospace vehicles.
- Climate modeling.
- Improving environmental decision making.
- Discovery of non-renewable energy sources.
- Understanding bio-molecular structures.
- For more details on these and other problems.
  - `http://www.nitrd.gov/pubs/200311_grand_challenges.pdf`
  - `http://ceee.rice.edu/Books/CS/chapter1/intro52.html`

Aerospace

- **Computational Aero-sciences Project**
  NASA - NASA Ames, NASA Langley and NASA Lewis
  - Accelerate the development and availability of high-performance computing technology that will be of use to the U.S. aerospace community, facilitate the adoption and use of this technology by the U.S. aerospace industry, and hasten the emergence of a viable commercial market for hardware and software vendors to exploit this lead

- **High performance computational methods for coupled field problems and GAFD turbulence**
  NSF - Colorado, Minnesota, and the National Center for Atmospheric Research
  - Develop and implement algorithms and software on parallel computers for solving field problems in structural and fluid dynamics and studying highly turbulent flows which arise in geophysical and astrophysical fluid dynamics.

Computer Science

- **High performance computing for learning**
  NSF - MIT, Brown and Harvard
  - Develop, implement, and test new mathematical techniques, software, and hardware for high performance computers with the ultimate goal of getting computers to “see, move, and speak”.

- **Parallel I/O methodologies for I/O-intensive Grand Challenge applications**
  NSF - Caltech and Illinois
  - Investigate and develop strategies for the efficient implementation of I/O intensive applications on a specially configured Intel Paragon computer. They will characterize I/O behavior and performance, define I/O models and methodologies, and develop, implement and test tools to support scientific applications with large I/O requirement.

Energy

- **Mathematical combustion modeling**
  DOE
  - Developing adaptive parallel algorithms for computational fluid dynamics and applying them to combustion models.

- **Numerical Tokamak project**
  DOE - Lawrence Livermore, Texas, UCLA, Oak Ridge, Princeton, NASA JPL, Cornell, Los Alamos, Caltech, National Energy Research Supercomputer Center
  - Develop and integrate particle and fluid plasma models on massively parallel machines as part of the multidisciplinary study of Tokamak fusion reactors.

Energy (contn"d)

- **Oil reservoir modeling**
  DOE - Texas A&M, Brookhaven, Oak Ridge, Rice, Stony Brook, South Carolina, and Princeton
  - Develop software for massively parallel computers that calculates fluid flow through permeable media. The project has a dual application, focusing on methods that solve modeling problems for petroleum reservoirs and for groundwater contamination.

- **Quantum chromo-dynamics calculations**
  DOE - Los Alamos
  - Developing lattice gauge theory algorithms on massively parallel machines for high energy physics and particle physics applications.

Environmental Monitoring and Prediction

- **Adaptive coordination of predictive models with experimental observations**
  NSF - Stanford and NASA Ames
  - Using a predictive computer model carrying out simulations in real time and a laboratory test bed, the team will investigate the potential for the interplay of the simulations and the experimental facility to estimate what data need to be gathered, as well as the location and resolution of this data, in order that accurate predictions of the future behavior of a complex nonlinear fluid system such as the atmosphere or the ocean can be made.

- **Computational chemistry**
  DOE - Argonne, Pacific Northwest Laboratory, Allied Signal, du Pont, Exxon, and Phillips
  - Develop new parallel algorithms, software, and portable tools for computational chemistry, and develop modeling systems for critical environmental problems and remediation methods.

Environmental Monitoring and Prediction (cont"d)

- **Data analysis and knowledge discovery in geophysical databases**
  NASA - UCLA, NASA JPL
  - Demonstrate the applicability of information systems for geophysical databases to support cooperative research in earth-science projects.

- **Development of algorithms for climate models scalable to TeraFLOP performance**
  NASA - NASA Goddard
  - Develop a high-resolution global climate model capable of centuries-long calculations on massively parallel machines at teraFLOP speed.

Environmental Monitoring and Prediction (cont"d)

- **Development of an Earth system model: atmosphere/ocean dynamics and tracers chemistry**
  NASA - UCLA, Princeton, Berkeley, Santa Barbara, JPL, Lawrence Livermore
  - Develop a model of the coupled global atmosphere-global ocean system, including chemical tracers that are found in, and may be exchanged between the atmosphere and the oceans. Use the model to study the general circulation of the coupled atmosphere-ocean system, the global geochemical carbon cycle, and the global chemistry of the troposphere and stratosphere.

- **A distributed computational system for large scale environmental modeling**
  NSF - Carnegie Mellon and MIT
  - Use high performance heterogeneous computing systems, advanced software environments, parallel architectures, and networks to develop algorithms for multiphase chemistry and aerosol dynamics and a distributed computing approach for simultaneous solution and sensitivity of environmental models.

Environmental Monitoring and Prediction (cont"d)

- **Earthquake ground motion modeling in large basins**
  NSF - Carnegie Mellon, USC and the National University of Mexico
  - Develop new mathematical models and software tools to demonstrate the capability for predicting, by simulation on parallel computers, the ground motion of large basins during strong earthquakes, and use this capability to study the seismic response of the Greater Los Angeles Basin.

- **Four-dimensional data assimilation for massive Earth system data analysis**
  NASA - NASA Goddard, NASA JPL, Syracuse
  - The goal of data assimilation is the calculation of consistent, uniform, spatial and temporal representations of the Earth environment that can be used for scientific analysis and synthesis. This involves the collection of diverse Earth observational data sets, and the incorporation of these data into models of the ocean, land surface, and atmosphere, including chemical processes.

Environmental Monitoring and Prediction (cont"d)

- **Global climate modeling**
  DOE - Los Alamos, Argonne, Oak Ridge
  - Numerical studies of the Earth's climate using general circulation models of the atmosphere and ocean.

- **Groundwater transport and remediation**
  DOE - Texas A&M, Brookhaven, Oak Ridge, Rice, Stony Brook, South Carolina, and Princeton
  - Develop software for massively parallel computers that calculates fluid flow through permeable media. The project has a dual application, focusing on methods that solve modeling problems for petroleum reservoirs and for groundwater contamination.

Environmental Monitoring and Prediction (cont"d)

- **High performance computing for land cover dynamics**
  NSF - Maryland, New Hampshire, Indiana, and NASA Goddard
  - Develop techniques to support access and analysis of remotely sensed data stored on parallel disk systems and use those techniques to facilitate the study of global ecological responses to climate changes and human activity.

- **Massively parallel simulation of large scale, high resolution ecosystem models**
  NSF - Arizona
  - Establish new algorithms and implementations for massively parallel processing that integrate geographical information systems databases with cellular discrete-event methodology to express large scale realistic ecosystem models and visualize their simulated behavior. The focus will be on monitoring and predicting landscape and ecosystem changes for large geographic regions

Molecular Biology and Biomedical Imaging

- **Advanced computational approaches to biomolecular modeling and structure determination**
  NSF - Illinois, Duke, NYU, Yale, and Eli Lilly Corporation
  - Develop models and molecular dynamics algorithms for a widely used program for structural biology (X-PLOR) in order to advance the fundamental understanding of molecular biology and pharmacology.
- **Computational biomolecular design**
  NSF - Houston
  - Use emerging scalable parallel computers and software to develop and implement new methods for solving critical problems in biomolecular design.

Molecular Biology and Biomedical Imaging (cont"d)

- **Computational structural biology**
  DOE - Caltech, Argonne, University of Washington, and UCLA
  - Understanding the components of genomes and developing a parallel programming environment for structural biology.
- **High performance imaging in biological research**
  NSF - Carnegie Mellon and Pittsburgh
  - Use the latest technologies in light microscopy and reagent chemistry with advanced techniques for computerized image analysis, processing and display, implemented on high-performance computers to produce an automated, high speed, interactive tool that will make possible new kinds of basic biological research on living cells and tissues.

Molecular Biology and Biomedical Imaging (cont"d)

- **Understanding human joint mechanics through advanced computational models**
  NSF - Rensselaer Polytechnic and Columbia
  - Develop automated and adaptive three-dimensional finite element analysis and parallel solution strategies to describe nonlinear moving contact problems characteristic of the biomechanics of joints in the human musculoskeletal system using the actual anatomic geometries and the multiphasic properties of the tissues in the joint.

Product Design and Process Optimization

- **First-principles simulation of materials properties**
  DOE - Oak Ridge, Brookhaven, NASA Ames
  - Investigate new methods for performing large-scale, first-principles simulation of materials properties using a hierarchy of increasingly more accurate techniques that exploit the power of massively parallel computing systems.

- **High capacity atomic-level simulations for design of materials modeling**
  NSF - Caltech, Columbia and NASA JPL
  - Formulate and implement new methodologies for parallel computers to carry out high capacity atomic-level simulations for design of materials, and apply the resulting software to critical industrial materials problems.

Space Science

- **Black hole binaries: coalescence and gravitational radiation**
  NSF - Texas, Illinois, Syracuse, Pittsburgh, Penn State, Northwestern, North Carolina and Cornell
  - Create a computational toolkit to provide modular development tools to support the study of coalescence of astrophysical black holes and the gravitational radiation emitted via the numerical solution of Einstein's equations for gravitational fields.

- **Convective turbulence and mixing in astrophysics**
  NASA - Colorado, Michigan State, Chicago, Argonne, NCAR
  - Develop the next generation of multi-dimensional hydrodynamic codes for astrophysical simulations involving turbulent convection, based on the use of massively parallel machines.

Space Science (cont"d)

- **Cosmology and accretion astrophysics**
  NASA - Los Alamos, Syracuse, Penn State, Caltech, Australian National University
  - Develop parallel, scalable particle codes (N-body, smoothed particle hydrodynamic (SPH), and hybrid) based on hierarchical tree data structures and use them to study astrophysical problems.

- **The formation of galaxies and large-scale structure**
  NSF - Princeton, Illinois, Pittsburgh, MIT, Indiana, San Diego
  - Explore different numerical algorithms, mesh adaptation strategies, programming models and new software technologies in order to obtain detailed numerical simulations that can help answer the question: "What is the origin of large-scale structure in the universe and how do galaxies form?"

Space Science (cont"d)

- **Large scale structure and galaxy formation**
  NASA - University of Washington, University of Toronto
  - Develop the tools needed for high performance N-body simulations, and use these to test the "standard model" for the origin of galaxies and large-scale structure by accurately evolving it into its present highly nonlinear state.

- **Radio synthesis imaging**
  NSF - Illinois, Wisconsin, Maryland, Berkeley
  - Implement a prototype of the next generation of astronomical telescope systems - remotely located telescopes connected by high-speed networks to very high performance computers and on-line data archives.

- **Solar activity and heliospheric dynamics**
  NASA - Naval Research Laboratory, NASA Goddard
  - Develop parallel algorithms for solar and heliospheric modeling.

How to tackle these problems?

- Faster algorithms
  - Faster sequential approaches to solving the problem
  - Parallel algorithms
- Faster Machines
  - Faster processors, memory and interconnection
  - Parallel machines
- Improvements in all these are necessary, but, in most cases
  Sequential algorithms running on single processor machines are not enough : need for parallel algorithms on parallel machines

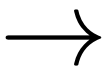
Why Sequential Architectures are not enough?

- Clock speeds are bounded by physical laws.
- Instruction level parallelism already exists in processors
  - Pipelining
  - Superscaler processors
  - VLIW (Very Long Instruction Word) Architectures

  But requires very complex hardware and/or sophisticated compilers.
- Vector processors work well only for certain types of problems.

High Performance Computing Initiative (HPCI)

Thesis: Only Scalable Concurrent Computing may solve the problems.



Parallel and Distributed Computing

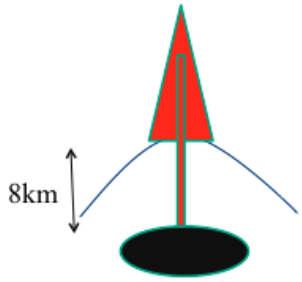
What is Parallel Processing?

What is Parallel Processing?

- The *art* of solving a single problem using multiple computational resources.
  - Decompose a problem into smaller subproblems that **can** be solved concurrently.
  - Solve the subproblems
  - Integrate the small problem solutions to form a solution to the large original problem.
- Typical decomposable problems:
  - Array computations (Vector, Matrix calculations)
  - PDEs, ODEs, Linear Systems Solution.
  - Transforms: Laplace, Fourier, Radon (CAT scanner).

Example: Oil Mantle

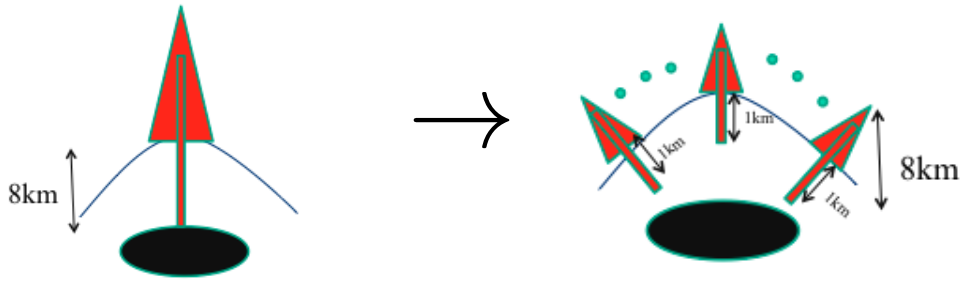
Suppose an oil mantle exists at a depth of 8 km.



- If the well is drilled at a rate of 1 km per day, the mantle can be reached in 8 days.
- Can we accelerate the process?

Example: Oil Mantle (cont"d)

Sure, drill 8 1-km wells, with the 8 wells, the mantle is reached in 1 day !!!



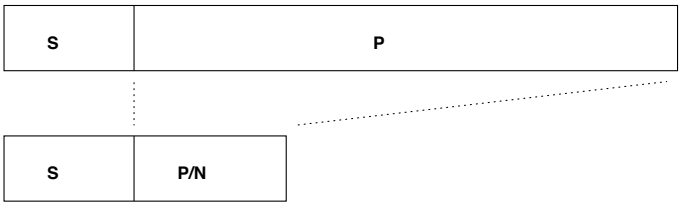
- Obviously wrong !!! ... are we solving the right problem ???
- Parallelizable problem: After 8 days, the oil can be extracted 8 times faster !!!

Expected Speedup

What speedup can we expect from Parallelism? (1)

Let Π be a program that contains two main parts:

- A sequential part of length s (units of time)
- A part that can be parallelized of length p



The speedup obtained by executing the program in an N-processor parallel computer is:

$$Speedup = \frac{s+p}{s+\frac{p}{N}}$$

What speedup can we expect from Parallelism? (2)

Normalizing with $s + p = 1$, s and p become the fraction of total unit time spent to execute the program.

$$Speedup = \frac{1}{s + (1-s)/N} = \frac{N}{Ns + 1 - s} = \frac{N}{1 + (N-1)s}$$

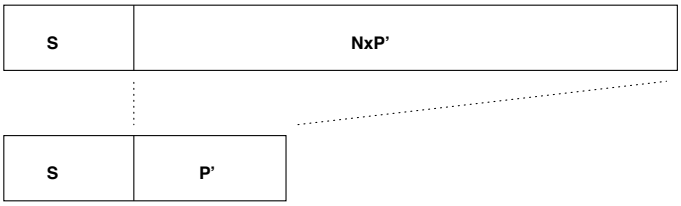
From the above expression, $Speedup \to \frac{1}{s}$ as $N \to \infty$

- This is known as Amdahl's law and it says that the best speedup one can expect is limited by the *sequential bottleneck* regardless of the number of processors.

Sooooooooo, WHY THE FUSS!

Amdahl's Law Revisited (1)

Consider now a very large problem where the parallel part is *sequentialized* to execute in a single processor.



The Slowdown obtained by executing the program on a single processor computer is:

$$Slowdown = \frac{s + Np'}{s + p'}$$

Amdahl's Law Revisited (2)

Normalizing with $s + p' = 1$, s and p' become the fraction of total unit time spent to execute the program.

$$Slowdown = s + Np' = s + (1 - s)N = N - s(N - 1)$$

or

$$Slowdown = (1 - p') + Np' = (N - 1)p' + 1$$

which is also the expression for Speedup when going from a single processor to an N-processor parallel system. Hence, from the above expression, $Slowdown \to N$ as $s \to 0$ or, conversely, $p \to 1$

THIS IS THE FUSS

- This is known as Gustafson's correction to Amdahl's law and it basically says that one must solve LARGE problems in large machines rather than the same little problems we were used to solve in little machines.

Characteristics of a Parallel Algorithm

- Do not measure SPEEDUP by comparing the Parallel Algorithm Time and its Sequentialized Time.
- Characterize the improvement as the best SPEEDUP attainable with respect to THE BEST SEQUENTIAL existing algorithm.

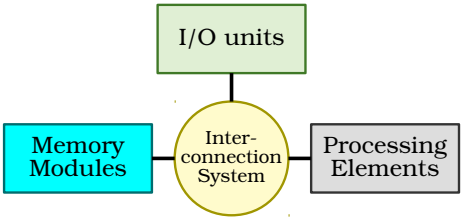
- The three main parameters to take into account:
  - **Comput. Complexity:** Number of basic operations.
  - **Memory:** Amount of Storage necessary.
  - **Communications:** Number of basic network operations

Architecture and Classification of Parallel Computers

Components of a Parallel Machine

- Processor
- Memory
- Interconnect (Processor-to-Processor and Processor-to-Memory)
- I/O
- Programming model and tools

Flynn's Taxonomy



After M. F. Flynn, *Some Computer Organizations and their Effectiveness*, IEEE Trans. on Computers, September 1972, pp. 948-960.

Two types of parallelism: Control and Data

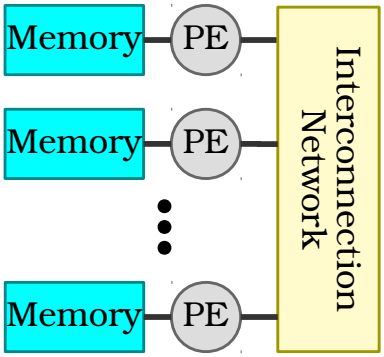
	Single Data Stream (SD)	Multiple Data Stream (MD)
Single Instruction Stream (SI)	SISD: Uniprocessor (VAX, IBM, RISC)	SIMD: Array Processors (MPPs, Associative)
Multiple Instruction Stream (MI)	Pipeline, Systolic Processors	MIMD: Multiprocessors CM-5, IPSC, Paragon

- SISD (Single Instruction, Single Data)
  - Single processor machines
  - Not really parallel!
- MISD (Multiple Instruction Single Data)
  - No real machine mapping.
- SIMD (Single Instruction Multiple Data)
  - Multiple processing units (PE) working in lockstep.
  - Same instruction executed by all PEs in each step on different data, single control unit to tell PEs what to do.
- MIMD (Multiple Instruction Multiple Data)
  - Multiple PE, two PE can be executing different instructions at the same time.

Parallel/Distributed Arch. - Instr./Data Classification

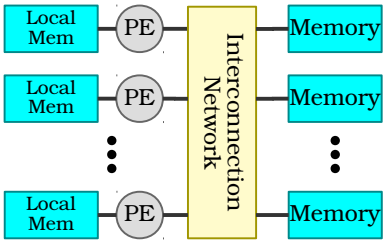
- MIMD Commercially available clusters.
- Preponderance of Data Parallel applications.
- New Parallel Programming Paradigm: Single Program Multiple Data (SPMD)

Basic Parallel/Distributed Architecture



- SIMD or MIMD Architecture.
  - Memory can be distributed (exclusive address space for each PE) or can be made look as shared (single address space for all PE)

Basic Parallel/Distributed Architecture (cont"d)



- MIMD Architecture. Some Local Memory and a Modular Shared Memory.
  - Local Memory can be shared (single address space for all PE) or distributed (different address space for different PE)

Examples

- SISD
  - mainframes, old workstations, old PCs.
- MISD
  - ...
- SIMD
  - **Shared Memory**: Hitachi S3600 Series
  - **Distributed Memory**: Cambridge Parallel Processing Gamma II Plus
- MIMD
  - **Shared Memory**: Cray J90/T90, DEC Alphaserver, SGI Origin 3000, modern multicore PCs.
  - **Distributed Memory**: Cray T3D/T3E, Cray XT3, plus recent workstation clusters (IBM SP2, DEC, Sun, HP).

For a good overview of architectural classes for HPC, see
`http://www.netlib.org/utk/papers/advanced-computers/`

General Model for Parallel Processing Architectures

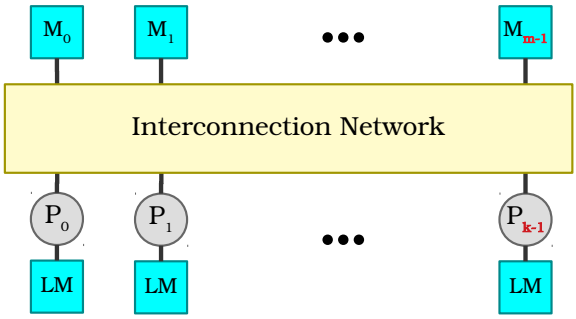
- The Single Program Multiple Data (SPMD) is the model of choice for Data Parallel programming.
- Concurrent Computers are hence built as MIMD machines with very powerful computing nodes.
- With the advent of Multicore microprocessors, computing nodes are also Parallel Computing engines in their own right.
- It is the Interconnection Network that gives rise to the different modern concurrent computing systems currently available.

MY QUEST:

Treat the Interconnection Network as a Co-Processor capable of contributing to the efficiency of concurrent computations.

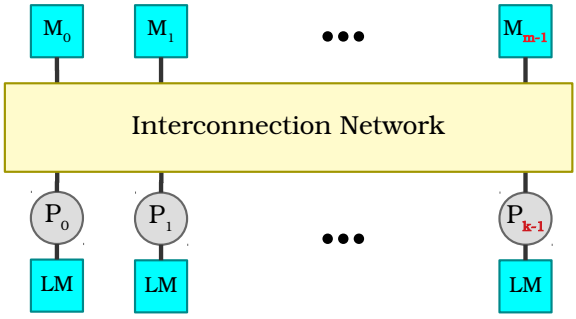
I SHALL SHOW YOU !!!

General Parallel Processing Architecture



- Number of memory modules (m) **not necessarily equal** the number of processing nodes (k).
- An interesting problem arises...

General Parallel Processing Architecture (Homework problem)



- Given a fixed number of computing nodes (say k), and assuming every computing node attempts to access a memory module on every network cycle, characterize the access time in network cycles for a typical computing node as a function of the number m of memory modules.
- Let us discuss !!!

Other Classifications and Programming Models

Other Classifications and Programming Models

Types of Parallelism:

- Control vs Data Parallelism
- Data Parallel and Single Program Multiple Data (SPMD)
- Coarse vs Fine grain Parallelism
- Shared Memory vs Message Passing (MPI)

Hardware Architectures

- Uniform Memory Access (UMA) or Symmetric MultiProcessors (**SMP**).
- Non-Uniform Memory Access (**NUMA**) and Cache Coherent Non-Uniform Memory Access (CC-NUMA)

Other Nomenclatures Used

- Massively Parallel Processors (**MPP**)
- Distributed Systems
- **Clusters**

SMP

- Shared memory MIMD
- Multiple processors of the same type
- Processors and memory connected to the same bus
- All processors are treated as equal, any task can be done by any processors
- Typical no. of processors less than 100 (typically 2-64)
- Task allocation to processors controlled by OS
- Most common OS's like Windows, Linux support SMPs

NUMA

- Each processor with its own physical memory
- Shared virtual address space
- Accesses to addresses in local memory faster
- Accesses to addresses in remote memory handled by hardware routers, slower
- Local cache at each processor
- Cache-coherency protocol needed to keep caches consistent (CC-NUMA)
- Small number of processors (less than 100)
- Examples – SGI Origin, Sequent NUMA-Q

MPP

- Specialized architectures with large number of processors (hundreds to thousands)
- Specialized fast interconnect switches to connect processors to processors and processors to memory
- Can be SIMD or MIMD, shared or distributed memory
- Examples – Cray T3D, MasPar's MP1/2, Thinking Machines CM1/2, CM5.

Clusters

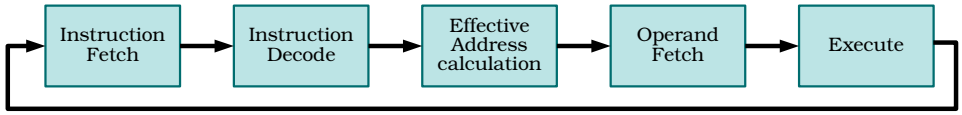
- Collection of interconnected **stand-alone** computers (nodes) that work together as a **single computing resource**.
- Front-end where tasks are submitted, allocated and load balanced among back-end machines transparently (Single System Image).
- Machines usually run same operating system kernel in each node.
- Distributed OS is challenging.
- Number of nodes can be from tens to thousands

Theoretical/Abstract programming models

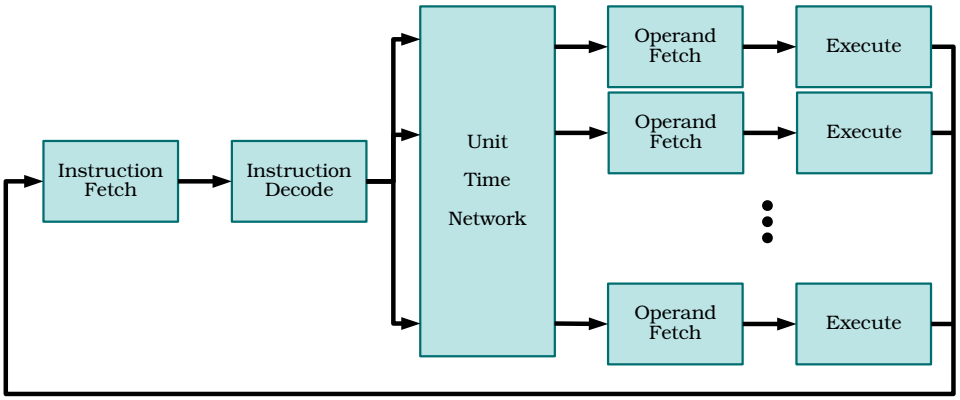
- **Parallel Random Access Memory** in its different forms: Concurrent/Sequential Read, Concurrent/Sequential Write: Processors gain conflict free access to a zero access time Shared RAM.
- **Bulk Synchronous Parallel Machine**: A BSP computer consists of components capable of processing and/or local memory transactions (i.e., processors), a network that routes messages between pairs of such components, anda hardware facility that allows for the synchronization of all or a subset of components.
- **LogP**: The LogP machine consists of arbitrarily many processing units with distributed memory. The processing units are connected through an abstract communication medium which allows point-to-point communication. This model is pair-wise synchronous and overall asynchronous. The machine is described by the four parameters:
  - L: the latency of the communication medium.
  - o: the overhead of sending and receiving a message.
  - g: the gap required between two send/receive operations. A more common interpretation of this quantity is as the inverse of the bandwidth of a processor-processor communication channel.
  - P: the number of processing units. Each local operation on each machine takes the same time ('unit time'). This time is called a processor cycle.

  The units of the parameters L, o and g are measured in multiples of processor cycles.

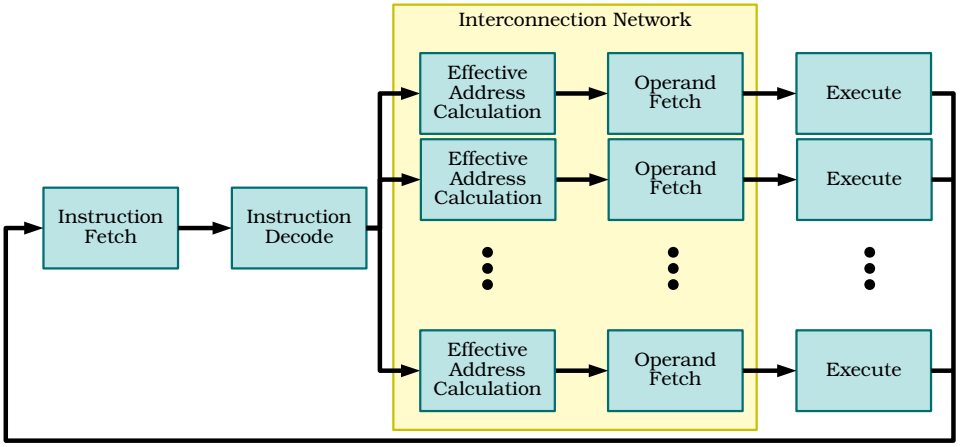
Von Neumann Cycle



Von Neumann PRAM



Von Neumann SIMD Machine



Von Neumann MIMD Machine