FORTRAN and MPI

Part 2: Message Passing Interface

DANISH CENTER FOR APPLIED MATHEMATICS AND MECHANICS

January 14 – 18, 2008

Plan of the course:

- Introduction - why parallel/HPC, aims, difficulties, terminology
- Parallel computers, physical and logical architecture
- Communication models
- Message Passing Interface (MPI) - keep it simple
- Communications in MPI (point to point)
- Parallel debugging tools
- MPI - more advanced features: datatype constructors.
- Linear algebra algorithms, operations with matrices and vectors
- MPI - further advances: communicators, virtual topologies
- Parallel sorting
- Parallel performance. Other parallel programming models
- Summary and tendencies

Day 1: Introduction

The need of high performance computing

*

In recent years the potential of available, serial as well as parallel, computer resources has been growing hand-in-hand with the appetite to solve numerically steadily larger models of real-life problems, both tendencies feeding on each other.

Grand challenges

*

The term *Grand Challenge* was coined in 1987 by prof. Kenneth G. Wilson. It characterizes the most demanding, pressing and difficult problems in computational science and engineering.

The list of Grand Challenge problems can never be complete since each area of science and engineering potentially poses new Grand Challenges.

Grand challenge problems

Computer simulations: HPC is enabling realistic simulations of many physical systems and the study of their complex behaviour.

Cheaper, faster, safer, more complex, size varying, more accurate than laboratory experiments.

Grand challenge problems

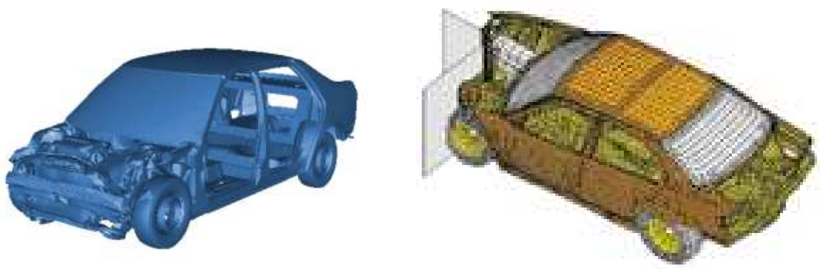
The majority of underlying problems are
highly interdisciplinary.

Physics and Chemistry:

- Molecular dynamics simulations, where the fundamental time step is around a femtosecond ($10^{-15} sec$)
- General Electro-Magnetic solvers
- CFD
- Quantum Chemistry, Exited states of molecules

Grand challenge problems: *Engineering*

- Design of new materials, including recording media and high-temperature superconductors
- Aerodynamic design of aerospace vehicles
- Microelectronic design, including the design of quantum switching devices
- Ignition and combustion modelling in automotive engine design
- Car crash modelling - FE dynamic responses of 3D inelastic structures

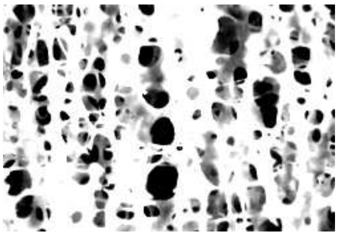


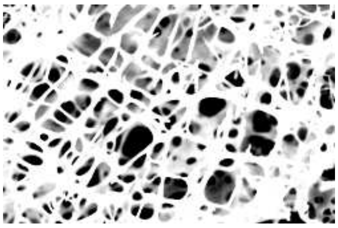
Grand challenge problems:

Computational medicine

- Forward and backward processes in x-ray and ultra-sound tomography
- $\surd$ Analysis of the mechanical behaviour of porous, trabecular bone structures and their dynamic adaptation
- Human circulatory systems and vascular diseases
- Rational design of new anti-cancer and anti-AIDS drugs
- Testing devices (for HIV, cancer)
- $\surd$ Genetic sequences
- Biocomputing

The 'Bone' problem



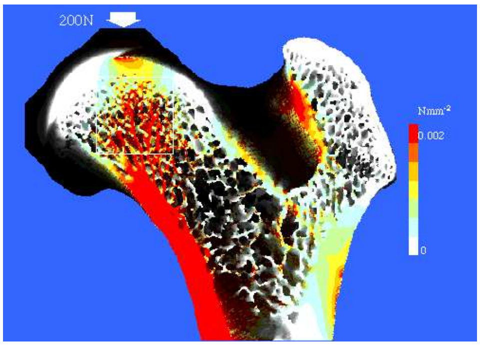


Claim: *The internal structure of biological tissues is "optimally designed" to the environment.*

Questions: (1) For which physical objective are bones optimal structures?

(2) How does the bone tissue accomplish the optimality of its architecture?

The ’Bone’ problem



In 2002: Voxels (3D pixels) $\approx$ 14 microns; $10^5 - 10^6$ FE elements per cm^3; Using micro-CT scanner, $418 \times 275 \times 385$ voxel grid, $7.6\,10^6$ FE elements, $1.8 Gbyte$, 20 CPU hours on Cray C90.

In 2005: M. Adams et al. *Ultrascalable implicit FE analyses in solid mechanics...*
... up to 537 million degrees of freedom on 4088 IBM Power3 processors (ASCII Blue) and an average time per linear solve of about 1 and a half minutes.

The ’Bone’ problem

In 2002: Voxels (3D pixels) $\approx$ 14 microns; $10^5 - 10^6$ FE elements per cm^3; Using micro-CT scanner, $418 \times 275 \times 385$ voxel grid, $7.6\,10^6$ FE elements, $1.8 Gbyte$, 20 CPU hours on Cray C90.

In 2005: M. Adams et al. *Ultrascalable implicit FE analyses in solid mechanics...*
... up to 537 million degrees of freedom on 4088 IBM Power3 processors (ASCII Blue) and an average time per linear solve of about 1 and a half minutes.

What has happened? Is the improvement in time due only to hardware advances?
In this case - no. There is also the influence of the numerical methods used, which in the second case are highly numerically efficient (Algebraic Multigrid).

Grand challenge problems

“Biology may well change the shape of High Performance Computing.”
Bioinformatics is more data intensive, much more about fast interrogation of data bases and comparison of long strings of data.
The measure is Giga-sequence comparisons per hour (GSC/hr).
BLAST (basic local alignment search tool) $\Longleftrightarrow$ BLAS, BLACS

- Genome sequencing algorithms

  How many genes are there? ≈ 35000? Or $120000 - 140000$?

  Genes are no more than the recipes that living cells “read” to manufacture proteins.
- Predicting the 3D structures of the proteins

  The proteins are the workhorses of biological systems.

Grand challenge problems - Celera database

Examples:
2002: 700 interconnected Alfa-64 bit processors, 1.3 GFLOPS, 50 terabyte database.
2006 (August): it contained over 65 billion nucleotide bases in more than 61 million sequences.
GenBank receives sequences produced in laboratories throughout the world from more than 100,000 distinct organisms. It continues to grow at an exponential rate, doubling every 10 months.

Grand challenge problems

Environment

- High-resolution weather forecasting: more accurate, faster and longer range predictions
- Pollution studies, including cross-pollutant interactions
- Global atmosphere-ocean-biosphere and long range climate modelling

  ⇒ Risø National Laboratory & Technical University of Denmark, Roskilde

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.
Task: establish reliable control strategies for the air pollution.

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.
Task: establish reliable control strategies for the air pollution.
Basic scheme: pollutants are emitted in the air from various sources

DEM - air pollution model

Danish Eulerian model: to perform numerical simulations and to study the air pollution over Europe.
Task: establish reliable control strategies for the air pollution.
Basic scheme: pollutants (many of them) are emitted in the air from various sources (many of them) and

- transported by the wind - (diffusion, advection, horizontal vertical)
- get deposited on the Earth surface
- transform due to chemical reactions
  - factors: winds, temperature, humidity, day/night, ...

DCAMM, Jan 14–18, 2008 Maya Neytcheva, IT, Uppsala University maya@it.uu.se – p.15/52

DEM - air pollution model

$$\frac{\partial c_s}{\partial t} + - \sum_{i=1}^{3} \frac{\partial u_i\, c_s}{\partial x_i} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left(K_{x_i}\frac{\partial c_s}{\partial x_i}\right) + E_s - (k_s c_s + Q(c_1, \cdots, c_q),$$

where $c_s, \quad s = 1, \cdots, q$ are the unknown concentrations of q species of the air pollutants to be followed.

DEM - air pollution model

Demands:

Spatial domain: 4800 km^2, 96×96 or 480×480 regular mesh (horizontal resolution)
10 vertical layers, 1 km each

35 pollutants

3D: $35 * 10 * 96^2 = 3225600$ unknowns
3D: $35 * 10 * 480^2 = 80640000$ unknowns
about 36000 time-steps, 'only one month simulated'

Computers at DCAMM: 16 PEs on Newton:

60 000 sec = 1000 min $\approx$ 17 h

Is this much or not?

Let us say something about

Supercomputers

HPC computers

“Supercomputers are the largest and fastest computers available at any point in time”.

This first definition of supercomputers is due to the *New York World*, March 1920, and is used in connection with *“new statistical machines with the power of 100 skilled mathematicians in solving even highly complex algebraic problems”*.

What was considered to be “super” yesterday
⇒ just ordinaries today
⇒ may even be forgotten tomorrow.

However, there has been a permanent strive to make the computers faster (in computing speed) and larger (in memory capacity), and the movement towards this major aim is a chain of impressive technological, architectural and algorithmic achievements.

Denmark in the Top500 list, November 2003

Rank	Manuf.	Computer	Installation Site	Year	# PEs	GFLOPS
1	NEC	Earth Simulator	Earth Simulation Center	2002	5120	35860
343	NEC	SX-6/64M8	Danish Meteorological Inst.	2003	64	459.2
481	HP	SP P2SC 160 MHz	Sonofon A/S	2003	192	408
500	IBM	xSeries Cluster Xeon 2.4 GHz - Gig-E	ShengriLi, China	2003	256	402

Denmark in the Top500 list, November 2005

Rank	Manuf.	Computer	Installation Site	Year	# PEs	GFLOPS
1	IBM	Blue Gene	DOE/NNSA/LLNL	2005	131072	280600
203	IBM	eServer 326 Cluster, Opteron 2.6 GHz, GigEthernet	DCSC, University of Copenhagen	2005	1024	2791
500	HP	SP Power3 375MHz	Sun Trust, Florida	2005	460	1645

UPPSALA UNIVERSITET

DCAMM

Denmark in the Top500 list, November 2007

Rank	Manuf.	Computer	Installation Site	Year	# PEs	GFLOPS
1	IBM	Blue Gene/L	DOE/NNSA/LLNL	2007	212992	478200
2	IBM	Blue Gene/P	Forschungszentrum Juelich	2007	65536	167300
496	IBM	xSeries x3455 Cluster Opteron, 2.6 GHz, GigEthernet	DTU	2007	2416	5949.4
500	HP	SP Power3 375 MHz	Semiconductor company, UK	2007	1344	5929.6

UPPSALA UNIVERSITET

DCAMM

Major aim of HPC

Exploit the power of the HPC technologies to obtain a far greater throughput of results.

HPC - aims and difficulties in achieving these

① Pay several million US$ (EUR) for a supercomputer

② Wait for scientists to port their software to the new machine

③ Watch the machine's manufacturer be sold, go bankrupt, or move to a different market

④ Repeat

Price-performance ratio

Some examples

A casualty of War: Cray's first supercomputer Cray-1 (1976), peak performance 133 MFLOPS. First installation at Los Alamos Nat. Lab.

2002: Beowulf cluster of PCs (Ingvar) at NSC Linköping; 33 990 MHz PCs connected via Ethernet; 26.05 GFLOPS.

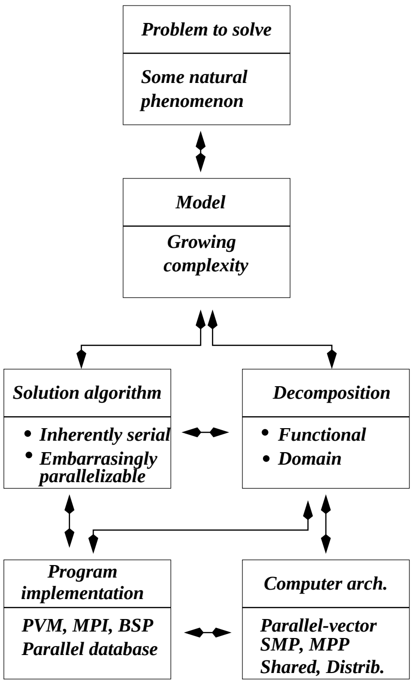
Today’s giants

The Earth Simulator

The future?

BlueGene/L is scaled up with a few unique components and IBM’s system-on-a-chip technology developed for the embedded microprocessor marketplace. The computer’s nodes are interconnected in three different ways instead of the usual one. Using a cell-based design, BlueGene/L is a scalable architecture in which the computational power of the machine can be expanded by adding more building blocks, without introduction of bottlenecks as the machine scales up.

Numerical Simulations - chain of steps



UPPSALA UNIVERSITET

DCAMM

HPC terminology

- Supercomputing
- HPC
- Parallel computing - HPC using multiprocessor machines
- Distributed computing - more added functionality than performance
- Grid Computing, Metacomputing - distributed HPC
- Internet computing (SETI project)

Homogeneous vs heterogeneous computer systems

UPPSALA UNIVERSITET

DCAMM

Principles of Parallel Computing

- Parallel and serial parts coexistence
- Granularity
- Locality
- Load balance
- Coordination and synchronization
- Performance modelling and measuring

UPPSALA UNIVERSITET

Granularity

The term *granularity* is usually used to describe the complexity and type of parallelism, inherent to a parallel system.
granularity of a parallel computer and *granularity of computations*

- fine grain parallelism; fine-grained machine;
- medium grain parallelism; medium-grained machine;
- coarse grain parallelism; coarse-grained computer system.

UPPSALA UNIVERSITET

Performance barriers (parallel overhead)

- Startup (latency) time
- Communication overhead
- Synchronization costs
- Imbalance of system resources (I/O channels and CPUs)

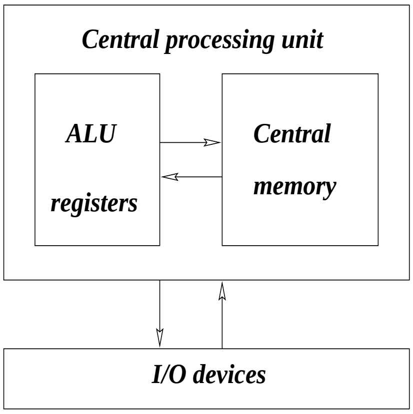
each of these can be in the range of milliseconds, i.e., millions of flops on modern computer systems

- Redundant computation
- load (dis-)balance

- Tradeoff: to run fast in parallel there must be a large enough amount of work per processing unit but not so large that there is not enough parallel work.

Computer architecture

The von Neumann architecture (since 1945)



- programs and data are treated in the same way; they both reside in the computer memory;
- the instructions are executed in a consecutive manner, following the order of how they have been programmed;
- conditional branches are allowed;
- during the execution, an information exchange between the CPU and memory takes place, involving data or program instruction;
- input of programs and data, as well as output of results is done by communicating to the outer world.

`fetch` → `decode` → `execute` → `store`

The von Neumann architectural principles are subject to a unique interpretation. It has been implemented in a great variety of computers which, although very different in performance, internal organization, technological base, etc, have much in common.

- Their behaviour is well determined.
- Their performance is relatively easy to analyze and to predict.
- The software is easily ported from one computer to another.

Taxonomy of the "non-von" computer architectures

Flynn's taxonomy

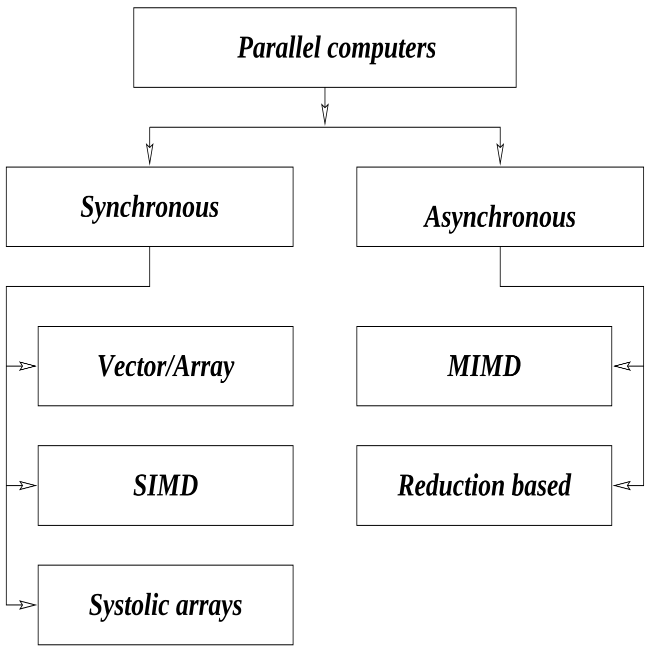
A classification from a programming point of view:

Instruction stream	Data Stream	
	SD	MD
SI	SISD	SIMD
MI	MISD	MIMD

* MISD - the empty class, kept for completeness * SPMD - missing in the table but frequently utilized

* MIMD - too broad by itself

Synchronous/asynchronous parallel computing



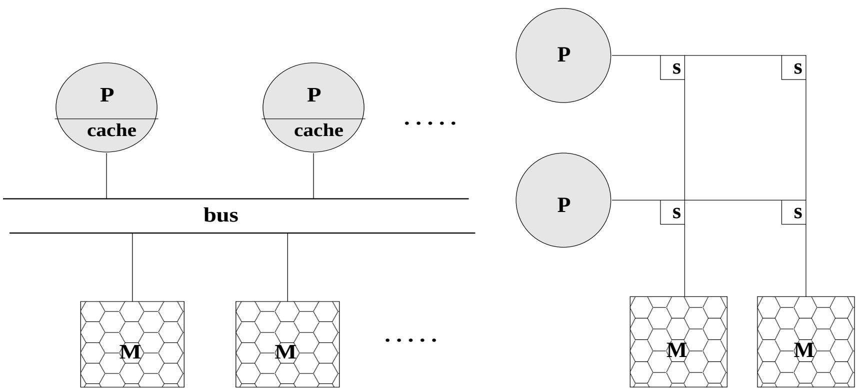
Taxonomy of the "non-von" computer architectures wrt memory

- Shared memory - SMPs
- Distributed memory
- Hybrid - clusters of SMPs with a fast interconnection network)
- Grids - distributed memory (heterogeneous systems) plus LAN, WAN, Internet

Taxonomy of the "non-von" computer architectures wrt memory

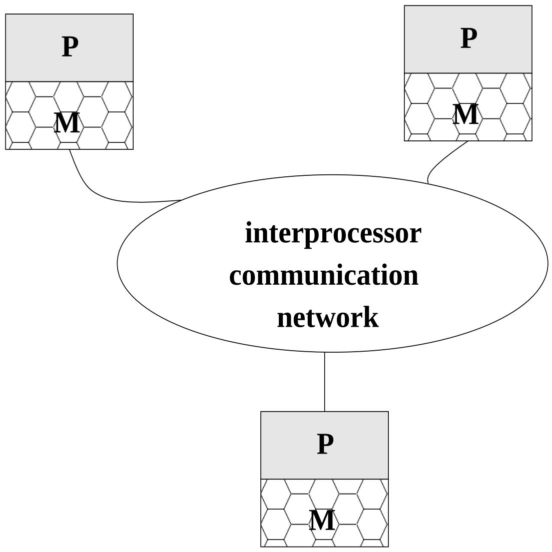
Shared/distributed memory machines

Shared memory machines

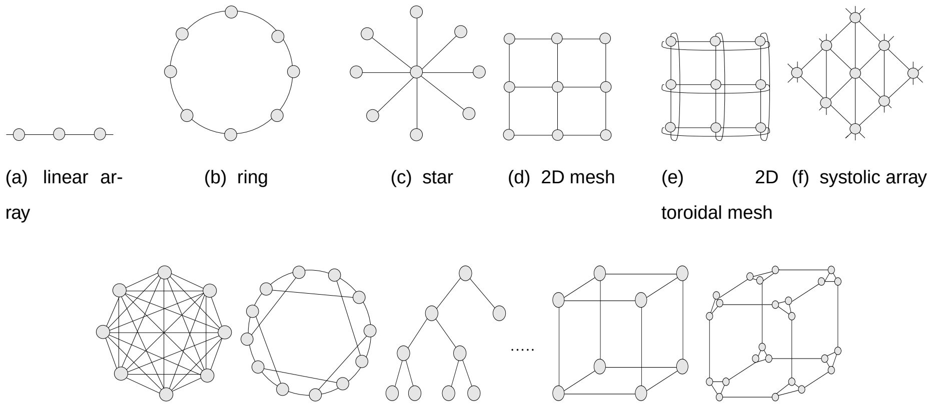


(a) bus-connected (b) crossbar-connected

Distributed memory machines



Interconnection network topologies

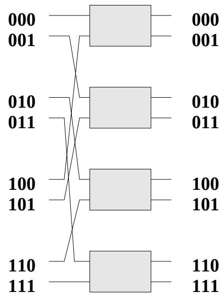


(a) linear array (b) ring (c) star (d) 2D mesh (e) 2D toroidal mesh (f) systolic array

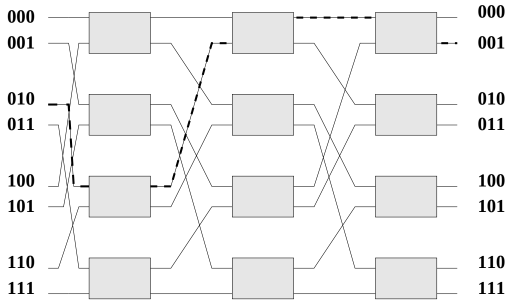
(g) completely connected (h) chordal ring (i) binary tree (j) 3D cube (k) 3D cube

UPPSALA UNIVERSITET

Dynamic interconnection networks



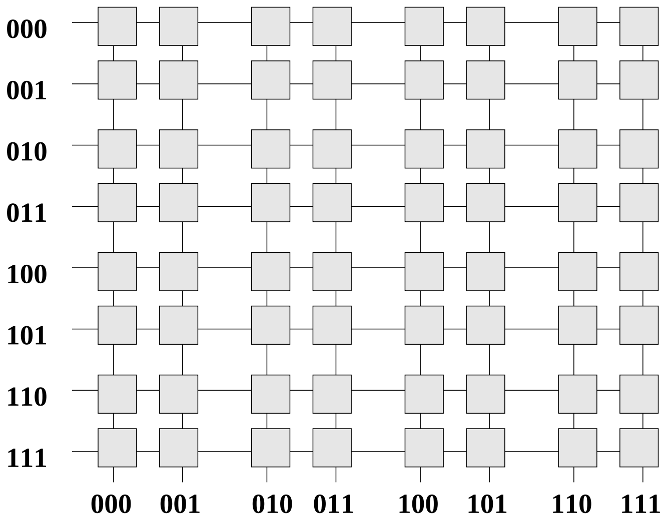
(a) Shuffle-Exchange



(b) Omega Network

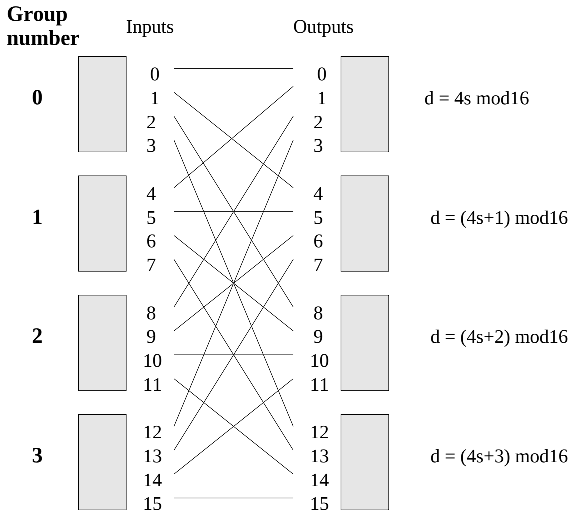
UPPSALA UNIVERSITET

Dynamic interconnection networks



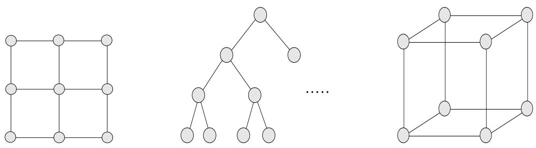
Crossbar switch

Dynamic interconnection networks



Perfect shuffle connection
between 16 PEs and 16 Memories

Recall:



We can impose different logical architecture over one and the same hardware using different paradigms.
Examples:

- Model shared memory as distributed and vice versa, distributed memory as shared
- Model different communication network topologies on a existing hardware interconnection network

The logical architecture is related to the algorithm.

Do we have a good *algorithm – machine* match?

Parallel programming environment

– provides
- language tools
- application programming interfaces (APIs)

A programming environment implies
a programming model (particular abstract model of the computer system)

The jargon of the parallel computing, cont.

task	a sequence of instructions that operate as a group
process	collection of resources, that enables the execution of program instructions
thread	associated with a process, shares the process resources
PE	a generic term for a hardware unit that executes a string of instruction

'Task' and 'Thread' are referred as 'Units of execution' (UEs)

The jargon of the ..., cont

load balance	refers to how well tasks are mapped to UEs, and UEs to PEs so that the work is evenly distributed among the PEs
synchronization	enforcing necessary event/computation ordering to ensure a correct result
	pros <–> cons, synchronous <–> asynchronous
deadlocks	a cycle of tasks where these block each other by waiting for the availability of a certain resource (not that hard to detect)
race conditions	one and the same program may produce different results with the same data (due to errors in synchronization; may be harder to detect)

Before writing a parallel program

we need to go through a number of design steps:

- find concurrencies
- determine the algorithm structure
- choose supporting structures
- utilize some implementation mechanism

Find concurrencies

- Decomposition – tasks or data (shared, distributed, arrays, recursive structures)
- Dependency analysis – group tasks, order tasks, data sharing (RO, RW, ...)
- Design evaluation – simplicity, flexibility, efficiency, suitability to a computer platform.

Algorithm structure

- Task organization
  - task parallelism
  - divide-an-conquer
- Data decomposition
  - geometric decomposition (DD)
  - recursive data patterns
- Data flow
  - pipeline pattern (regular)
  - event-based coordination (irregular)

Simplicity, flexibility, efficiency.

Supporting structures

- Program structures – SPMD, Master/Worker, loop parallelism, fork/join
- Data structures – shared, distributed

Implementation

- Task/process management
- Synchronization
- Communications

MPI

MPI – the standard programming environment for distributed memory parallel computers.

Logical/abstract distributed memory computer systems are meant.
MPI could run on a PC as well as on a shared memory computer, where the distributed memory is simulated by the software.