Introduction to Parallel Computing

Sofien GANNOUNI
Computer Science
E-mail: gnnosf@ksu.edu.sa ; gansof@yahoo.com

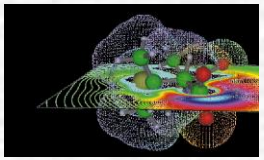
Parallelism

- **Parallel Computing:**
  - is a fundamental technique by which computations can be accelerated.
  - is a form of computation in which many calculations are carried out simultaneously.
- **Parallel Computers:**
  - classified according to the level at which the hardware supports parallelism:
    - **multi-core** and **multi-processor** computers having multiple processing elements within a single machine.
    - **clusters**, **Massively Parallel Processing**, and **grids** use multiple computers to work on the same task.
- **Parallel Programming:**
  - Decomposing a programming problem into tasks
  - Deploy the tasks on multiple processors and run them simultaneously
  - Coordinating work and communications of those processors

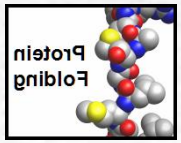
Why do we need Parallel Processing?

- Sequential machines are reaching their speed limits.
- Use multiple processors to solve large problems faster.
- Microprocessors are getting cheaper and cheaper.
- Cheap multiprocessors and multi-core CPUs bring parallel processing to the desktop.
- Many applications need much faster machines

Challenging Applications

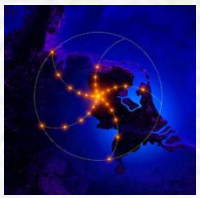


- Modeling ozone layer, climate, ocean
- Quantum chemistry
- Protein folding



- General: *computational science*

- Aircraft modeling
- Using volumes of data from scientific instruments
  - astronomy
  - high-energy physics



- Computer chess
- Analyzing multimedia content

History

- 1950s: first ideas (see Gill's quote)
- 1967 first parallel computer (ILLIAC IV)
- 1970s programming methods, experimental machines
- 1980s: parallel languages (SR, Linda, Orca), commercial supercomputers
- 1990s: software standardization (MPI), clusters, large-scale machines (Blue Gene)
- 2000s: grid computing: combining resources world-wide (Globus), General Purpose-GPU

Opportunities to use parallelism

- **Instruction Level Parallelism**
  - Hidden Parallelism in computer programs
- **Single computer level**
  - Multi-core computers: Chip multi-processors
    - Dual-core, Quad-core, GP-GPU
  - Multi-processor computers: Symmetric multi-processors
    - Super-computers
- **Multiple computers level**
  - Clusters, Servers, Grid computing

Parallelism in Computer Programs

- **The main motivation for executing program instructions in parallel is to complete a computation faster.**
- **But programs are written assuming that:**
  - Instructions are executed in **sequential**.
  - Most programming languages embed sequential execution.

Instruction Level Parallelism (IPL)

- **(a+b) * (c+d)**
  - could be computed simultaneously.
- **Separation of instructions and data.**
  - Instructions and memory references execute in parallel without interfering.
- **Instruction Execution is pipelined**
- **Processors initiate more than one instruction at a time.**

Parallel Computing vs. Distributed Computing

- **Parallel computing**
  - Provides performance that a single processor can not give.
  - Interaction among processors is frequent
    - Fine grained with low overhead
    - Assumed to be reliable.
- **Distributed Computing**
  - Provides:
    - Availability, Reliability
    - Physical distribution, Heterogeneity
  - Interaction is infrequent,
    - Coarse grained - heavier weight
    - Assumed to be unreliable.

Aspects of Parallel Computing

- **Parallel Computers Architecture**
- **Algorithms and applications**
  - Reasoning about performance
  - Designing parallel algorithms.
- **Parallel Programming**
  - Paradigms
  - Programming Models
  - Programming languages
  - Frameworks
  - Dedicated environments