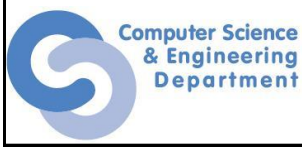
Parallel Programming

SCPD Master Module

Emil Slusanschi
emil.slusanschi@cs.pub.ro
University Politehnica of Bucharest



Acknowledgement

- The material in this course has been adapted from various (cited) authoritative sources by Lawrence Rauchwerger from the Parasol Lab and myself and is used with his approval
- The presentation has been put together with the help of Dr. Mauro Bianco, Antoniu Pop, Tim Smith and Nathan Thomas from the Parasol Lab at Texas A&M University

Grading@PP

- Activity during lectures – 1 point
  - Presence in class for the lectures is **compulsory** but does not insure the point – you have to (try to) participate **actively**
- Project work – 5 points
  - Similar to APP:
    - 3/coding, 1/documentation, 1/presentation, *1/bonus*
  - Topics from subjects related to the PP
  - Teams of 2-3 people – independent grading
  - Subject can also be done in the "research" hours – at the end a paper/presentation should emerge
- Oral exam – 4 points
  - 5-10 minutes / person
  - 2-3 subjects from the lecture
  - Can be **replaced** by holding a talk during the semester on a topic agreed with me in advance

"Your" Feedback

- Last year's feedback:
  - Important because you learn how to present ideas in precise and accurate English
  - Team work experience is vital
  - Working on big SW projects is important
  - Improvements required:
    - **Hard** rules & deadlines
    - Periodic evaluation of **individual** effort
    - Learn during the whole year (3 week session)
    - Put emphasis on reading…
    - Individual oral exam

Deadlines

- Choosing the Project:
  - Soft-deadline 31.10
  - Hard-deadline 7.11
- Project Status
  - Agreed at the lab by each team
- Project Submission
  - The only Deadline: 4.01.2011
  - Project Presentations on 4.01 & 11.01

Project Work Roadmap

- One page project description (pdf + trac) due 7.11.2010
  - **Introduction:** A one paragraph description of the significance of the application.
  - **Description:** A one to two paragraph brief description of what the application really does
  - **Objective:** A sentence on what I would like to accomplish with the team on the application – we have to agree on this at the lab.
  - **Background :** Outline the technical skills (type of Math, Physics, Chemistry, etc) that one needs to understand and work on the application.
  - **Resources:** A list of web and traditional resources that students can draw for technical background, general information and building blocks. Give URL or trac/svn links. Only use our trac/svn.
  - **Contact Information:** Name, e-mail, group and master program the team members are part of.
- The labs on 14.10, 21.10, 28.10, 4.11 are dedicated to presentation of project ideas by you and me and used to recruit teammates

Table of Contents (subject to change)

- Introduction to Parallelism
- Introduction to Programming Models
- Shared Memory Programming
- Message Passing Programming
- Shared Memory Models
- PGAS (Parallel Global Address Space) Languages
- Other Programming Models

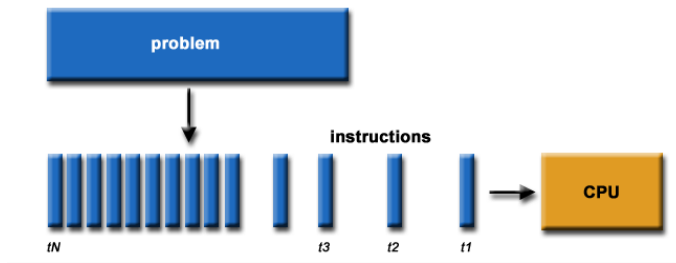
What Will You Get from this Lecture

- (New) Ideas about parallel processing
- Different approaches to parallel programming
- Various programming models used in parallel computing
- Practical experiences with some of the models/technologies presented in this lecture

Table of Contents

Introduction to Parallelism

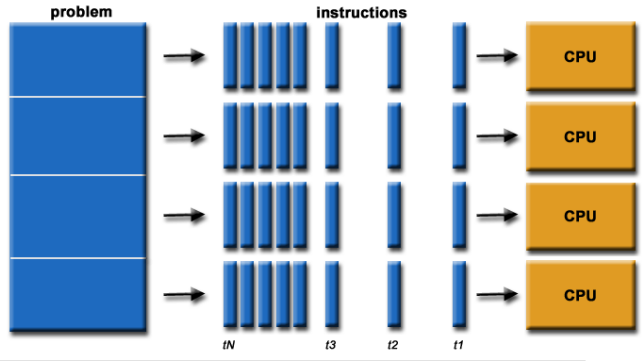
- Sequential Computing
  - Single CPU executes stream of instructions.



Adapted from: http://www.llnl.gov/computing/tutorials/parallel_comp

Introduction to Parallelism

- Parallel computing
  - Partition problem into multiple, concurrent streams of instructions.



Classification

Flynn's Taxonomy (1966-now)		Nowadays
SISD *Single Instruction* *Single Data*	**SIMD** *Single Instruction* *Multiple Data*	**SPMD** *Single Program* *Multiple Data*
MISD *Multiple Instructions* *Single Data*	**MIMD** *Multiple Instructions* *Multiple Data*	**MPMD** *Multiple Program* *Multiple Data*

- Execution models impact the above programming model
- Traditional computer is SISD
- SIMD is *data parallelism* while MISD is pure *task parallelism*
- MIMD is a mixed model (harder to program)
- SPMD and MPMD are less synchronized than SIMD and MIMD
- SPMD is most used model, but MPMD is becoming popular

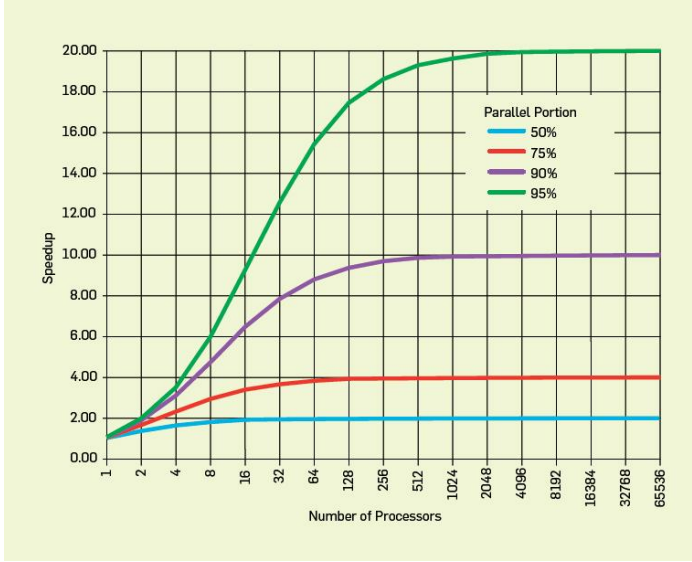
Introduction to Parallelism

- Goal of parallel computing
  - Save time - reduce wall clock time.
    - Speedup = $\frac{\text{wall-clock time of serial execution}}{\text{wall-clock time of parallel execution}}$
  - Solve larger problems - problems that take more memory than available to 1 CPU.

Reduce wall clock time

- Methods
  - Parallelizing serial algorithms (parallel loops)
    - Total number of operations performed changes only slightly
    - Scalability may be poor (Amdahl's law)
  - Develop parallel algorithms
    - Total number of operations may increase, but the running time decreases
- Work Complexity
  - Serialization: parallel algorithm executed sequentially Serializing parallel algorithm may lead to sub-optimal sequential complexity

Amdahl's law revisited



Performance Models

- Abstract Machine Models (PRAM, BSP, and many, many others)
  - Allow asymptotical analysis and runtime estimations
  - Often inaccurate for selecting the right implementation/algorithm on a given architecture
- Programming Primitives Behavior
  - Allow the selection of the right implementation
  - Increases programming effort

Abstract Machine

- PRAM (Parallel RAM, shared memory)
  - Processors access a shared flat memory
  - Performing an operation or accessing a memory location has cost = 1
- BSP (Bulk Synchronous Parallel, distributed memory)
  - Computation proceeds through supersteps
  - Cost of a superstep is $w+h/g+l$
  - w is the time for computing on local data
  - h is the size of the largest message sent
  - g and l are architectural parameters describing network bandwidth and latency, respectively