Parallel Programming for High-Performance Applications

Henri Bal
Vrije Universiteit Amsterdam

Ana Varbanescu
University of Amsterdam
Overview

• What is parallel programming?
• Why do we need parallel programming?
• Organization of this course
• Practicum Parallel Programming
Parallel Programming

- Sequential programming
  Single thread of control
- Parallel programming
  Multiple threads of control
- Why parallel programming?
  Eases programming? Not really.
  Performance? Yes!
Famous quote

• “Parallel programming may do something to revive the pioneering spirit in programming, which seems to be degenerating into a rather dull and routine occupation”

• S. Gill, Computer Journal, 1958
Why do we need parallel processing?

- Many applications need much faster machines
- Sequential machines have reached their speed limits
- Memory becomes a bottleneck
  - DRAM access times improve only 10% per year
  - Caches more and more important
Moore’s law (1975)

- Circuit complexity doubles every 18 months
- Exponential transistor growth, resulting in exponential growth of processor speeds
- We’ve now hit a “power wall”, halting speed increases
- Transistor growth now is used for multicore CPUs, i.e. parallel machines!
Parallel processing

• Use multiple processors to solve large problems fast
  – Also increases cache memory & aggregate memory bandwidth
• Also called High Performance Computing (HPC)
• Multicore CPUs bring parallel processing to the desktop
• Can use many (inexpensive) multicore machines to build very large parallel systems
• Very cheap Graphics Processing Units (GPUs) also contain many simple cores and can be used for many parallel applications
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 worldwide (Globus)
• Now: multicores, GPUs, Cloud computing
Large-scale parallel machines

- Many parallel machines exist
- See [http://www.top500.org](http://www.top500.org)
- Current #1: Tianhe-2 (3,120,000 cores)
- Many machines use GPUs
Our DAS-4 cluster
Challenging Applications

- Modeling ozone layer, climate, ocean
- Quantum chemistry
- Protein folding
- General: *computational science*
- Aircraft modeling
- Handling use volumes of data from scientific instruments
  - Lofar (astronomy)
  - LHC (CERN, high-energy physics)
- Computer chess
- Analyzing multimedia content
- Generating movies
Pixar’s "Up" (2009)

Whole movie (96 minutes) would take 94 years on 1 PC
(4 frames per day; 1 second takes 6 days; 1 minute per year)
Application areas

• Engineering and design
• Scientific applications
• Commercial applications (transactions, databases)
• Embedded systems (e.g., cars)

• This class focuses on scientific applications
Applications we work on now

• Signal processing for LOFAR telescope
  – Example: searching pulsars is extremely difficult, need to guess the direction, distance, and rotation frequency
  – Prepares for SKA telescope (2018)
    • 10-100 x global internet traffic per year,
    • exascale processing

• Image analysis, digital forensics

• Global climate modeling
  – Understand future local sea level changes
  – High resolution→huge compute power
Enlighten Your Research Global award

#7 STAMPEDE (USA)

#10 EMERALD (UK)

CARTESIUS (NLD)

KRAKEN (USA)

SUPERMUC (GER)
About this Course

Goal: Study how to write programs that run in parallel on a large number of cores.
Focus on programming methods, languages, applications
Focus on distributed-memory (message passing) machines and new machines (GPUs)

Prerequisites:
- Some knowledge about sequential languages
- Little knowledge about networking and operating systems
- “Concurrency & Multithreading” useful but not required
Aspects of Parallel Computing

Algorithms and applications

Programming methods, languages, and environments

Parallel machines and architectures
Course Outline

- Introduction in algorithms and applications
- Parallel machines and architectures
  - Overview of parallel machines, trends in top-500, clusters, many-cores
- Programming methods, languages, and environments
  - Message passing (SR, MPI, Java)
  - Higher-level language: HPF
- Applications
  - N-body problems, search algorithms
- Many-core (GPU) programming (Ana Varbanescu)
- LOFAR software telescope (Rob van Nieuwpoort)
Course Information

Examination
  Written exam based on:
  - Reader: available electronically from Blackboard
  - Lectures

More information (and slides + example exam):
Practicum Parallel Programming

Separate practicum (6 ECTS)

- Implement algorithms in C/MPI and Java/Ibis
- Implement GPU algorithms
- Test and measure the programs on our DAS-4 cluster
More information on Practical

- mCS / mAI students can do Practical only in period 2
- mPDCS students can do Practical only in period 3
  - No (re)submissions outside the given period
- One kickoff meeting in first week of both periods
- Details will be put on Blackboard