Programming Large-scale Parallel Systems

Henri Bal
Vrije Universiteit Amsterdam

Clemens Grelck
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 (except for inherently parallel applications)
  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
- Cheap Graphics Processing Units (GPUs) contain many simple cores and can be used for many applications
  - New course “Programming Multi-core and Many-core Systems” (Period 4)
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)
• Now: multicores, GPUs, Cloud computing
Large-scale parallel machines

- Many parallel machines exist
- See http://www.top500.org
- Current #1: Sunway TaihuLight (10,649,600 cores)
- Many machines use GPUs
Our DAS-4 cluster
Our DAS-5 cluster (June 2015)
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, SKA (astronomy)
  - LHC (CERN, high-energy physics)
- Computer chess
- Analyzing multimedia content
- Generating movies
- Deep Learning
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)

**13 THE COMPUTER DATA IS “RENDERED”**

Rendering is the act of translating all of the information in the files that make up the shot—sets, colors, character movement, etc.—into a single frame of film. Pixar's Renderfarm is a huge computer system that interprets the data and incorporates motion blur. Each frame represents 1/24 of a second of screen time and takes about six hours to render, though some frames have taken as many as ninety hours.
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

• Machine learning, semantic web
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

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, top-500, clusters
  DAS
• Programming methods, languages, and environments
  Message passing (SR, MPI, Java) + HPF
• Applications
  N-body problems, search algorithms
  Climate modelling on distributed supercomputers
• Higher-level language: Chapel
  By Clemens Grelck – last 4 lectures
Course Information

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

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

Separate practicum (6 ECTS)

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

• Can do the practical either in period 2 or period 3
  – Same assignments
• No (re)submissions outside the given period
• One kickoff meeting in first week of both periods
• Recommendations:
  – mCS / mA1 students: period 2
  – mPDCS students / mComputational Science: period 3
    • Advice: attend kickoff meeting in November anyway
• Details will be put on Canvas
Optional practical work for this course

- You are encouraged to try out MPI after it has been discussed in class (after approximately 1 month)
- Use your own (multicore) laptop or get an account on DAS-4 through Kees Verstoep (c.verstoep@vu.nl) from your official (VU/UvA) student account
- (Limited) guidance from PPP supervisor Rutger Hofman (rutger@cs.vu.nl)
- Assignments:
  - http://mpitutorial.com/ (tutorial)
  - Ask Rutger for more