UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Lecture 1: Introduction

Parallel Programming (INF-3201)
University of Tromsø

John Markus Bjørndalen

Instructors

		Office	Email
John Markus Bjørndalen	Instructor	REALF A247	jmb@cs.uit.no
Edvard Pedersen	Assistant		gmtired@gmail.com

What is Parallel Programming?

Sequential programming:
- Instructions executed one by one
- Only one at any point in time

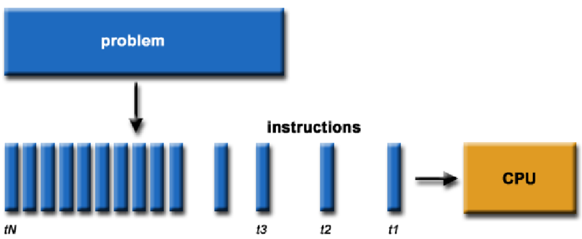


Figure from Blaise Barney, "Introduction to Parallel Computing", Livermore Computing.

What is Parallel Programming?

Parallel programming
- Break program into smaller parts
- Execute instructions from each part in parallel

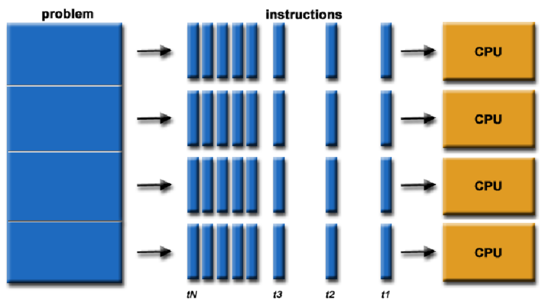


Figure from Blaise Barney, "Introduction to Parallel Computing", Livermore Computing.

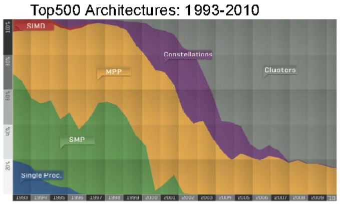
Why use Parallel Programming?

Save time and money

Parallel clusters from cheap, commodity components.

GPUs (many cores) can solve problems in real-time that we can't compute fast enough on a single core.

Top500 Architectures: 1993-2010



http://top500.org/statistics/overtime/

Why use Parallel Programming?

Solve larger problems
- Impractical to solve on a single computer
  Limited
  - Memory
  - CPU
  - Data/storage

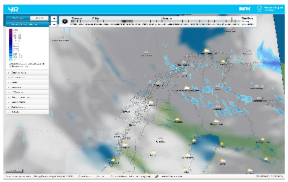
https://en.wikipedia.org/wiki/Supercomputer#/media/File:IBM_Blue_Gene_P_supercomputer.jpg

Why use Parallel Programming?

Solve larger problems

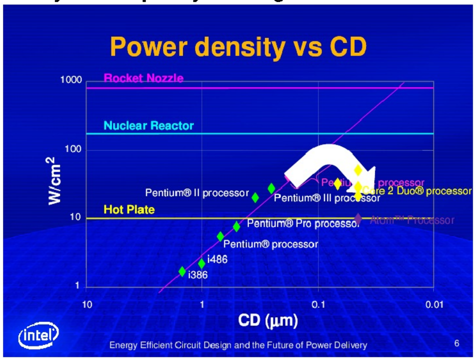
Examples:

- Search engines
- Big data
- Weather forecasting
- Particle research



Why use Parallel Programming?

Chip density and frequency: no longer «free lunch»



Energy Efficient Circuit Design and the Future of Power Delivery, Greg Taylor, EPEPS 2009

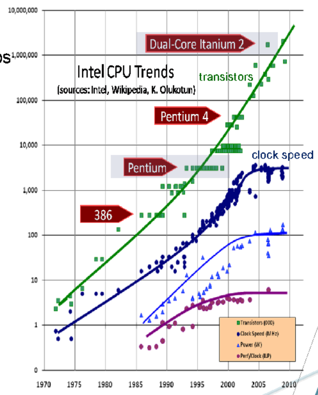
Scaling after hitting the wall

Clock speed hits a wall due to fundamental physics

Frequencies beyond 5GHz melt chips

The free lunch is over

Moore's Law: transistor density doubles every 18 months



Source: Herb Sutter, "The Free Lunch Is Over"

Scaling after hitting the wall

How to increase chip performance?

Chip density is still increasing

⇒ Can increase #cores/chip to increase chip performance and reduce power

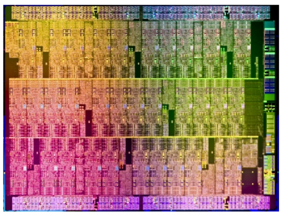


Image from http://www.itproportal.com/2012/08/01/a-close-look-at-xeon-phi-intels-50-core-beast/

Goals of the course

- Provide students with theory and practice on parallel programming
  - Theory
    - Fundamental concepts of parallel programming
    - Issues of parallel performance
    - Parallel programming techniques
    - Parallel algorithms
  - Practice
    - Parallel programming languages, environments and toolboxes
    - Hands-on experience with multi-core processors, graphics processors (GPU) and clusters.
    - Benchmarking and analyzing parallel programs

Theory

- Fundamental concepts of parallel programming
  - Parallel computers
    - message-passing multicomputer, shared memory multiprocessor, classification (Flynn's taxonomy), …
  - Parallel programming models
    - message-passing (clusters), shared memory (Intel multicore CPU), data-parallel model (Nvidia GPU), …
- Issues of parallel performance
  - Parallelism (Amdahl's law, DAG model), race-conditions, deadlocks, …
- Parallel programming techniques
  - Divide-and-conquer, load balancing, data-parallel exploitation, …
- Parallel algorithms
  - Sorting, numerical algorithms, prefix sums, …

Example: Matrix multiplication

Sequential program

```
for i = 1 to n do
  for j = 1 to n do
    x[i,j] = 0
    for k = 1 to n do
      x[i,j] = x[i,j] + A[i,k] * B[k,j]
```

⇒ execution time $O(n^3)$

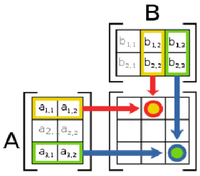


Illustration source: Wikipedia

$X_{n,n} = A_{n,n} \cdot B_{n,n}$

$x_{1,2} = a_{1,1}b_{1,2} + a_{1,2}b_{2,2}$

$x_{3,3} = a_{3,1}b_{1,3} + a_{3,2}b_{2,3}$

Example: Matrix multiplication

Sequential program

```
for i = 1 to n do
  for j = 1 to n do
    x[i,j] = 0
    for k = 1 to n do
      x[i,j] = x[i,j] + A[i,k] * B[k,j]
```

⇒ execution time $O(n^3)$

Parallel program

Different $X_{i,j}$s are independent

⇒ 2 outer loops can be parallelized

⇒ execution time $O(n)$

(with some assumptions – can you spot them?)

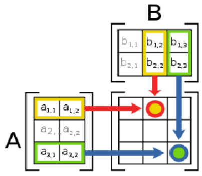


Illustration source: Wikipedia

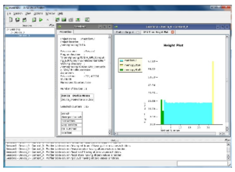
$X_{n,n} = A_{n,n} \cdot B_{n,n}$

$x_{1,2} = a_{1,1}b_{1,2} + a_{1,2}b_{2,2}$

$x_{3,3} = a_{3,1}b_{1,3} + a_{3,2}b_{2,3}$

Practice

- Parallel programming languages, environments and toolboxes
  - Message-passing model: Open MPI, an interface (API) & tools for developing large scale message-passing applications
  - Shared memory model: OpenMP, pragma based extension to C for improving performance on multi-core processors
  - Data-parallel model: CUDA/OpenCL, environments & tools for leveraging the massively parallel processing power of GPUs



CUDA Visual Profiler

NB: we will be making some changes to the projects this year. You will get more information later.

Projects

- You are given sequential codes
  - Parallelize code
  - Execute on cluster, multi-core computers and graphics processors (GPUs)
  - Evaluate results (speedup vs. expected)
  - Explain WHY!
- 3 mandatory (P1,2,3)

Course schedule

Monday	Friday
14:15 - 16:00 Lecture, REALF B203	08:15 – 10:00 Colloquium, REALF A016
	10:15-11:00 Lecture, REALF B203

- Assessment
  - 3 mandator assignments (pass/fail)
  - 1 final written exam: 100%

Important messages

- Make sure you have
  - Access to the course room in Fronter https://fronter.com/uit/
    - Schedules, handouts, important messages, hand-ins,...
    - Note that plans are tentative
  - Access to the gitlab repository:
    - https://source.uit.no/ifi-courses/inf-3201-2015-resources/tree/master
    - it should be public/open, so you don't need to log in
    - There is a link to the gitlab repository in the Fronter room (under archives).

References

- Blaise Barney, “Introduction to Parallel Computing”, Livermore Computing.
- Charles Leiserson et al., “How to Survive the Multicore Revolution”, Cilk Arts.
- Saman Amarasinghe, 6.189 “Multicore Programming Primer”, January (IAP) 2007. *(MIT OpenCourseWare).* http://ocw.mit.edu *(accessed 21 07, 2010).* License: Creative Commons Attribution-Noncommercial-Share Alike.
- Guy Blelloch & Bruce Maggs. “Parallel Algorithms” (from Computer Science Handbook, 2nd Edition, ISBN-13: 978-1584883609)