Overview

- Introduction
- Course Objective
- Themes
- Tools
- Material
- Assignments and Evaluation
- General Info

Overall Course Objective

- Our objectives and goals for you is that after this course you will
  - have gained an understanding for how supercomputers work
  - be able to apply several parallelisation techniques to solve scientific problems with a supercomputer

- The only way you can really gain this experience is learning-by-doing. The course is problem-driven based around a number of central concepts, and you will get your hands dirty with a small “cluster computer” available at ERDA – the UCPH Electronic Research Data Archive

Themes covered in the course

- The course will cover three distinct ways to speed up computation using a high performance server, a cluster, or a real bona-fide supercomputer
  - Vector processors and massively parallel procesors (GPUs)
  - Physical shared memory machines and threading
  - Distributed Remote Memory Machines and real clusters

- This progression of concurrency reflects the historical development of supercomputers starting with the dominance of CRAY vector computers and ending with upcoming exascale computers where all three levels of concurrency becomes important.

- In addition we will spend the very last week of the course considering the related challenge of parallelizing data access across machines.

Tools

- All tools will be available in time (we hope!)
- We will use C++ for programming
- Other aspects that will be covered are
  - Vector and GPU programming: OpenACC
  - Threading: Pthreads, OpenMP
  - Distributed computing and I/O: MPI – message passing interface

```cpp
#include <vector>
#include <array>
#include <cstdint>
#include <cmath>
#include <iostream>
#include <random>
#include <H5Cpp.h>
#include <chrono>
#include <argparse.hpp>

const double G = 6.673e-11;
const double SOLAR_MASS = 1.98892e30;

/** Class that represent a body, which can be a sun, a planet, or an asteroid */
class Body {
public:
    // The mass of the body
    double mass;
    // The position of the body in the x, y, and z axis
    double pos_x;
    double pos_y;
    double pos_z;
    // The velocity of the body in the x, y, and z axis
    double vel_x;
    double vel_y;
    double vel_z;
};

/** Class that represent a solar system, which consist of a sun, some planets, and many asteroids. */
class SolarSystem {
public:
    // The first Body is the sun and the rest are planets
    std::vector<Body> sun_and_planets;
    std::vector<Body> asteroids;
};

/** Function that returns -1 when `x` is negative, 1 when `x` is positive, and 0 when `x` is zero. */
double sign(double x) {
    if (x < 0) {
        return -1;
    } else if (x > 0) {
        return 1;
    } else {
        return 0;
    }
}

/** Function that returns `x` squared */
double squared(double x) {
    return x * x;
}

```

- Why C++ ?? We could have chosen Fortran, maybe Python, Julia, or commercial languages like MatLab or IDL.
- Everybody has the same initial conditions
- But! it doesn't matter. What matters are the principles. We will keep it simple.
- Important: OpenACC, OpenMP, and MPI are open standards available for the other languages.

Literature and additional material

- Links will be posted for each module of the course for
  - Notes
  - Papers
  - Books in PDF format

- Main material is covered in the lectures

- Main understanding comes from completing the assignments

Assignments

- There will be seven programming assignments, one for each phase of the course.
- In addition, you will get the chance to warm up you C++ skills this week
- All assignments will be fun! and related to a widely different set of physics examples

Assignments

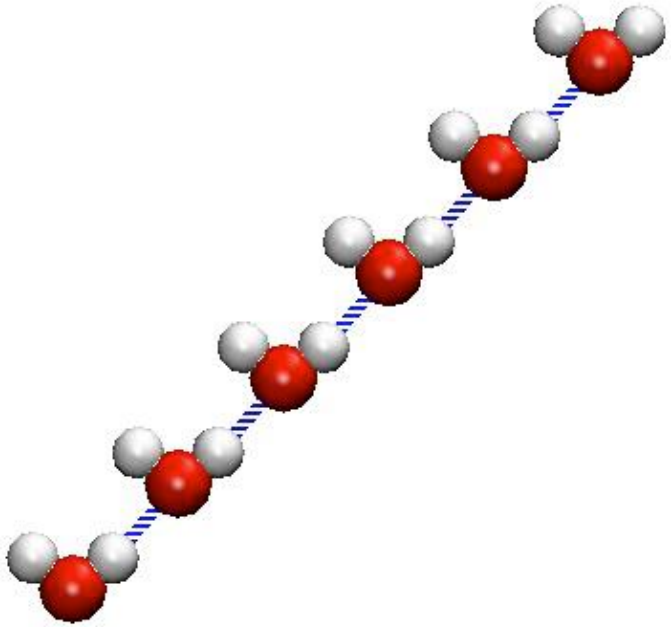
- This is a course about cluster computing!!

- For each assignment you will be given a "sequential implementation" of the problem. Something that works!
- Your job is to "parallelize it". This includes benchmarking and demonstrating speedup.

- Assignments must be documented and a report (with a page limit; important with 30 students) has to be submitted by **Sunday midnight** together with your code and a pdf "print" of the code.
- Final grade is average of the five best of the six assignments. All assignments have to be passed.
- Organize in groups of 2 or 3 for the weekly projects, and 4 to 6 for the final project

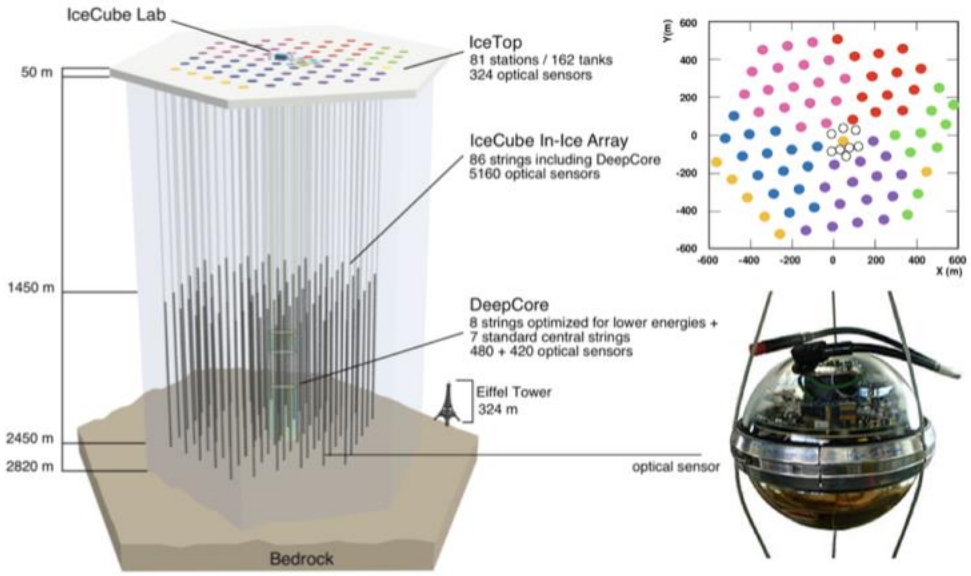
Assignment 2: Molecular dynamics (vectorization)

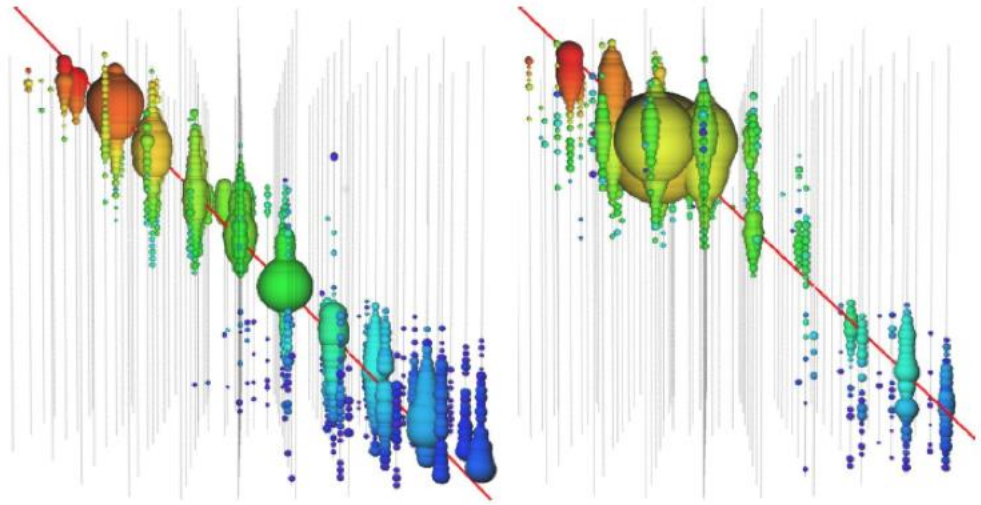
- Molecular dynamics allow us to model interacting molecules
- An example of the generic problem of "particle dynamics" – useful for describing molecules, fluids etc
- In the assignment you will model N interacting water molecules in a small volume



Assignment 3: Ice-cube detector (task farming)

- An important HPC application is the analysis of massive data sets or (Monte-Carlo) generation of simulated data points / events
- This can be accomplished efficiently using task farming
- You will work with data related to the ice-cube experiment



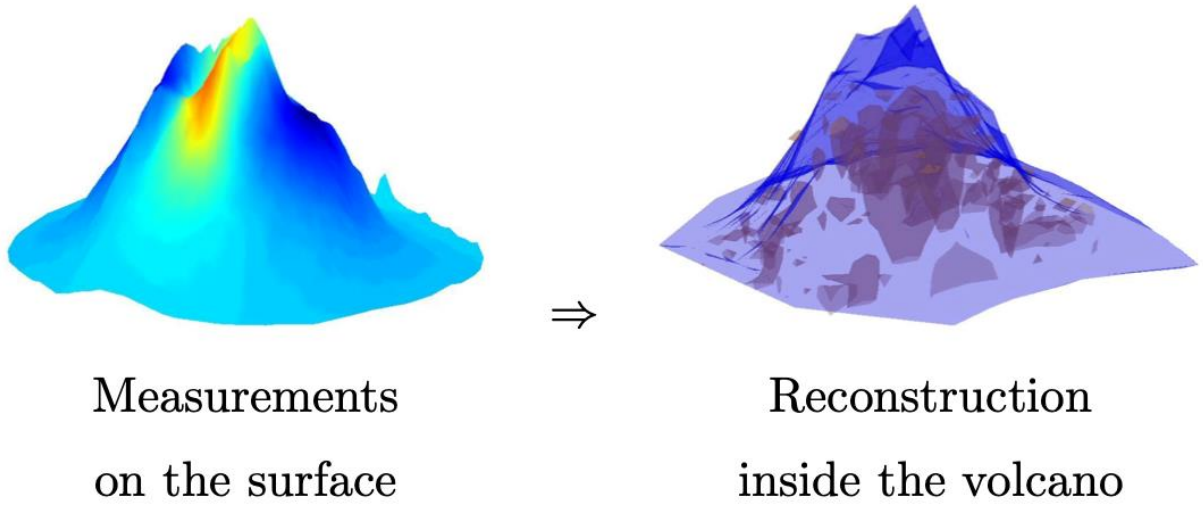


Assignment 4: Inverse problems (shared memory)

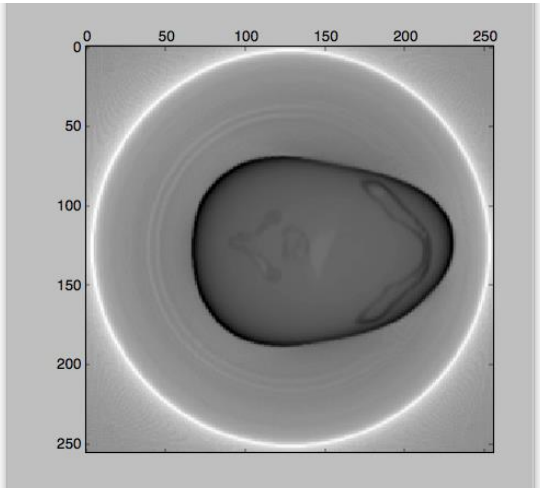
- Ill-posed problems arise in a variety of applications:
  - Seismology
  - Medical imaging (CT-scans)
  - Image deblurring (astronomy)
  - More general: removal of instrument window function
- Leads to an integral og elliptic constraint equation

$$\int_{\Omega} \text{input} \times \text{system}\, d\Omega = \text{output}$$

- Strongly coupled problem, and therefore best solved with shared memory parallelisation



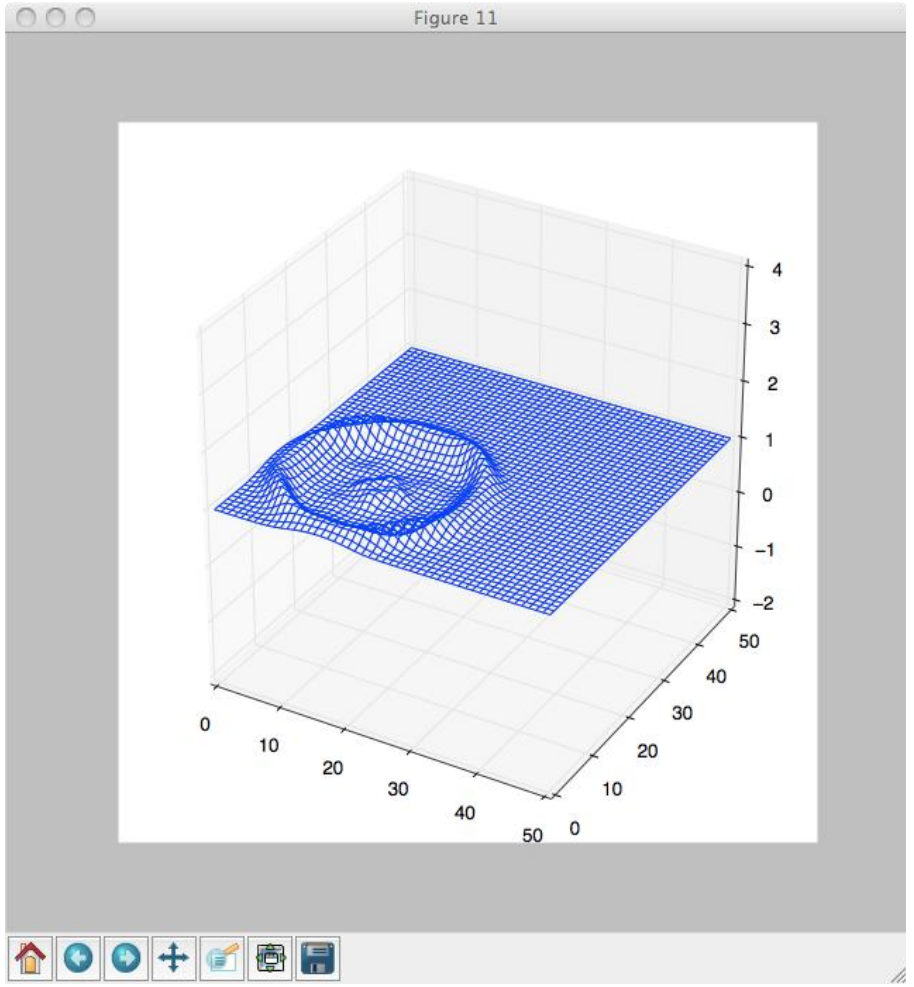
Measurements on the surface ⇒ Reconstruction inside the volcano



CT head scan

Assignment 5: Shallow Water (GPUs)

- Fluid dynamics
- A very common computational kernel design used in many science application
- “easy” to scale and speedup with parallel computation
- Regular execution on a grid well matched to GPUs



Assignment 6: Radiative transfer and Atmospheres (MPI)

- Atmosphere's are important for
  - weather prediction: short-term precise evolution
  - climate physics: long-term trends
  - stars and planets (structure and thermodynamics
- Classic "Stencil Computation"
- Pattern of computation seen in a large range of applications (transport of fluids and radiation through diffusion)