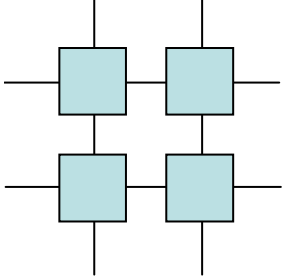
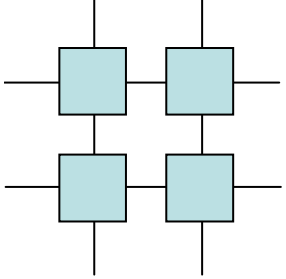


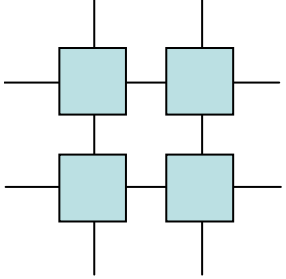
Introduction to Cluster Computing

Brian Vinter
vinter@diku.dk



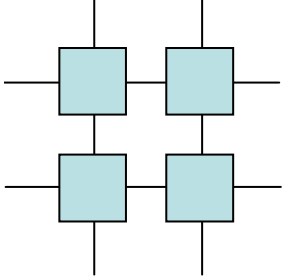
Overview

- Introduction
- Goal/Idea
- Phases
- Mandatory Assignments
- Tools
- Timeline/Exam
- General info



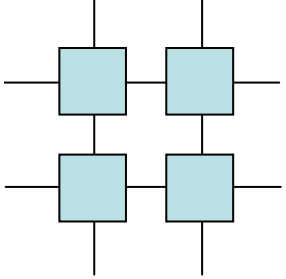
Introduction

- Supercomputers are expensive
- Workstations are cheap
- Supercomputers from workstations are cheap
  (but they may be hard to program)
- Several approaches to this problem exist



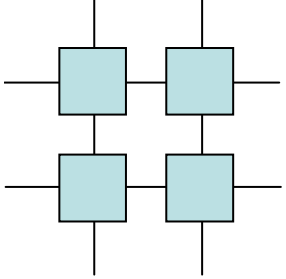
Goal/Idea

- The goal is that after this class you will be able to solve supercomputing class problems using a 'cheap' cluster solution
- The idea is that the only real way to learn this is by solving a set of problems - using a cluster



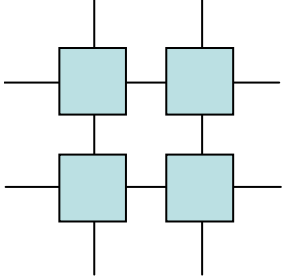
Approach

- Model classic supercomputer architectures to clusters
- Port the basic software from the SC architectures
- Emulate the hardware that might be missing



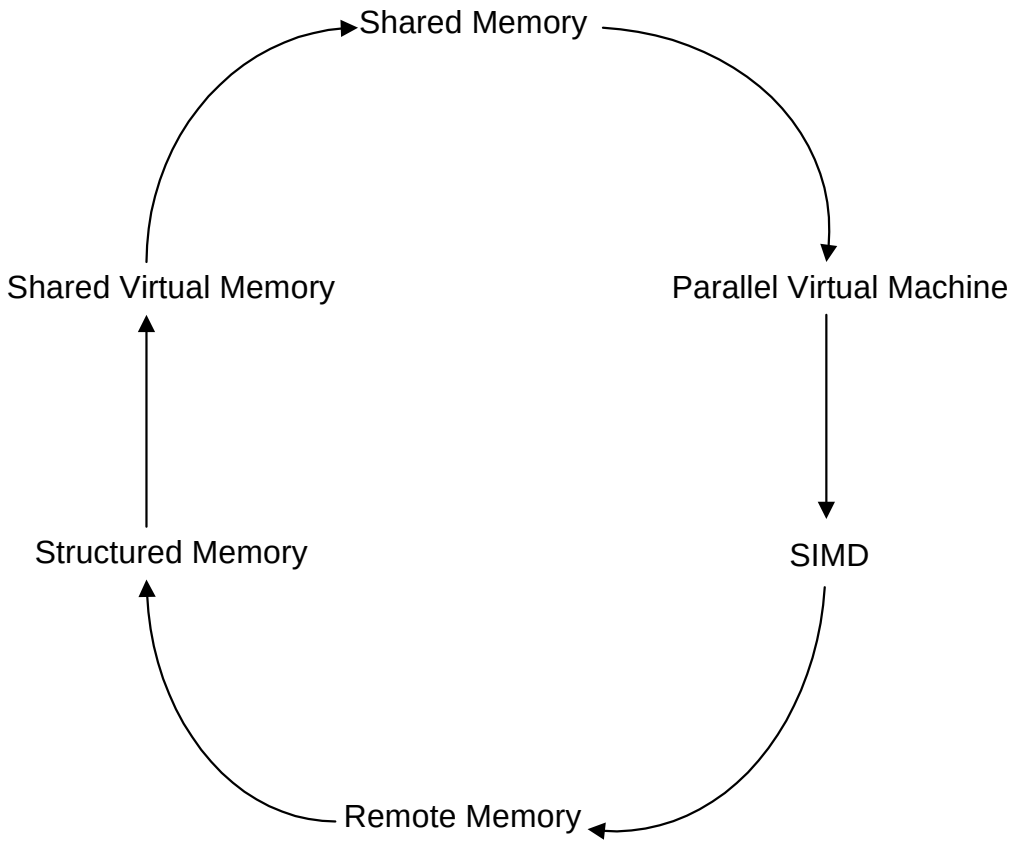
Phases

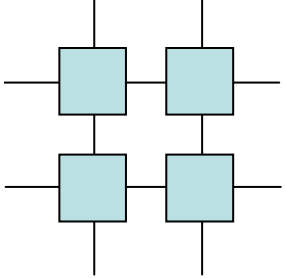
- The class will cover five distinct approaches to cluster computing:
  - Physical Shared Memory (not a cluster actually)
  - Abstract Machines
  - Emulated Massively Parallel Processors
  - Emulating Remote Memory Machines
  - Distributed Shared Memory



Cluster Computing

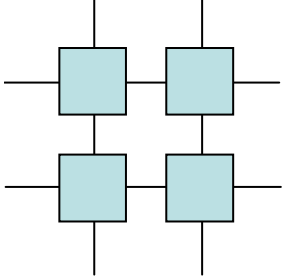
Evolution





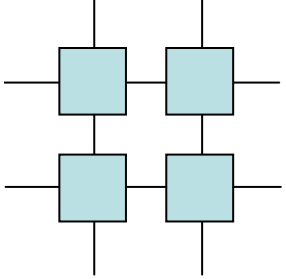
Tools

- All the tools will be available in time (I hope)
- We will use Java and C for programming
  - In fact the choice of language is free but one language will be recommended for each step
- Other tools that will be covered are:
  - Parallel Virtual Machines
  - Message Passing Interface
  - Remote Method Invocation
  - Tspaces, PastSet and TMem



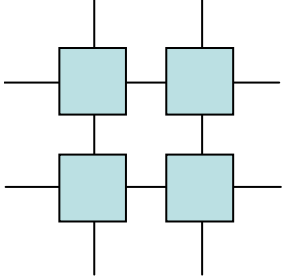
Tools

- Most tools are available in identical or similar form for use with other languages
- All the tools should be considered experimental and problems should be reported to me ASAP



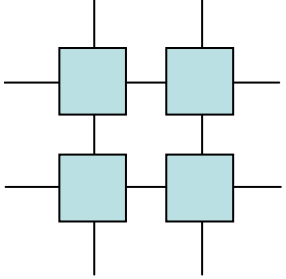
Litterature

- Notes
- Papers
- I can suggest books as they are needed ☺



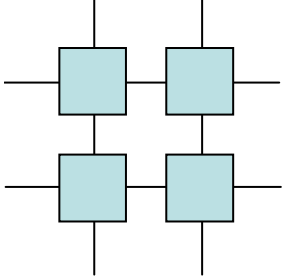
Mandatory Assignments

- There will be five (5) programming assignments, one for each phase we cover
  - In addition there may be a “paper only” assignment
- All assignments will be fun!

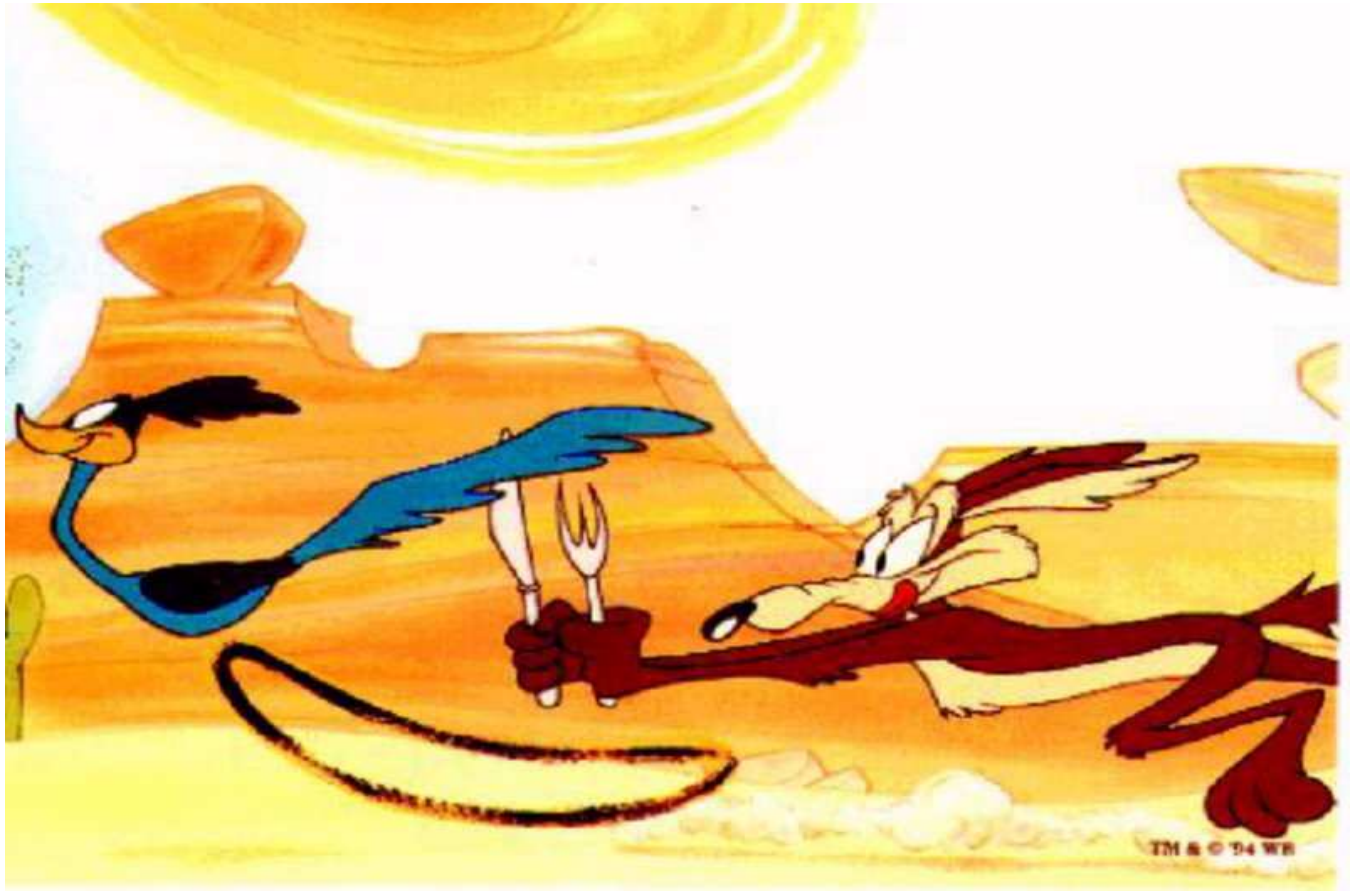


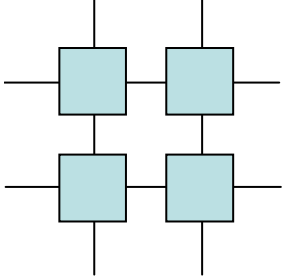
Assignments

- This is a class on cluster computing, thus the assignments will be accompanied by a sequential version of the problem, that you can work from
- Assignments must be documented and a report (there will be a page limit) submitted with the code
- In the end you only need to hand in 3 assignments
  - Plus perhaps a paper assignment



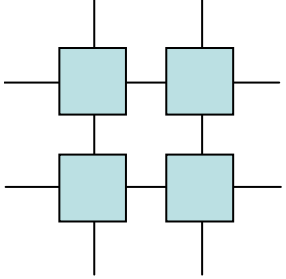
Assignments





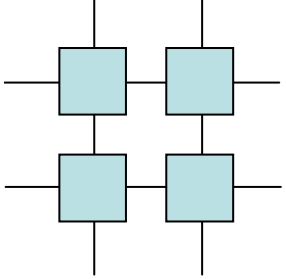
General Idea

- We cannot hope to implement one SC application within one class – let alone five!
- Most SC applications are based on a computational kernel which is fairly small and which takes up as much as 99.999...% of the total runtime
- You will port sequential verisons of such kernels to run on clusters
- To make it all fun we simulates cartoon traps

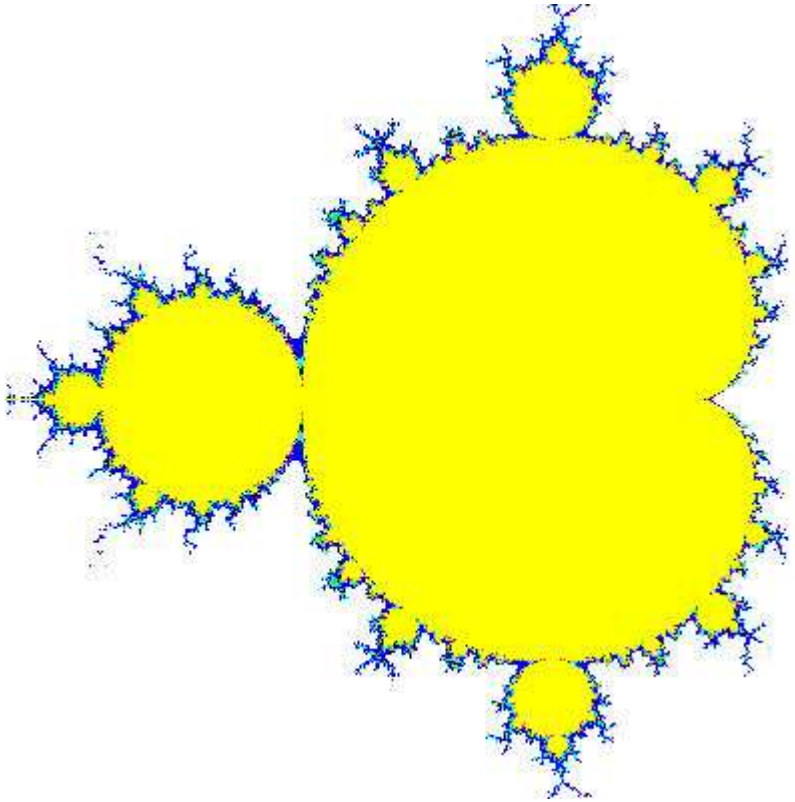


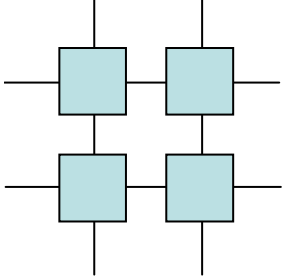
Road Map

- Fractals are examples of applications of the type we call embarrassingly parallel
- A typical example of an compute intensive application with many independent sub-results
- Very simple to write
- Achieves very good speedup



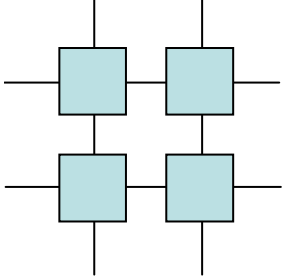
Road Map





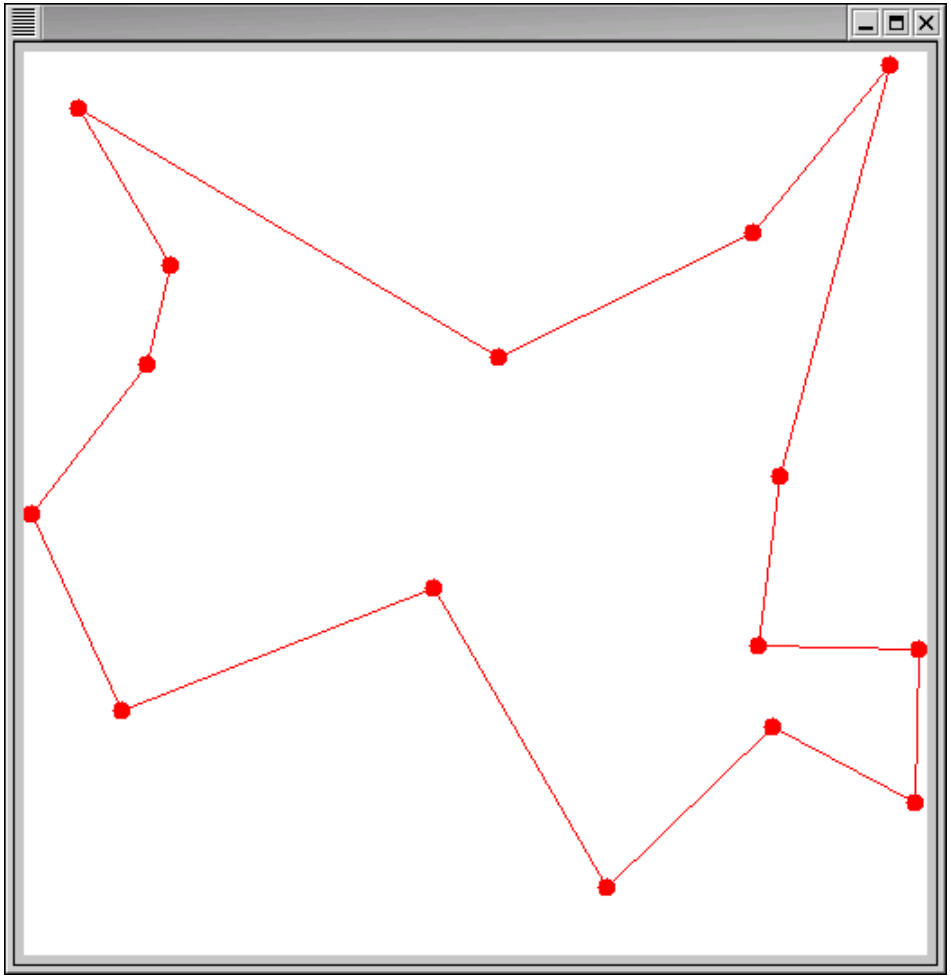
Race Trap

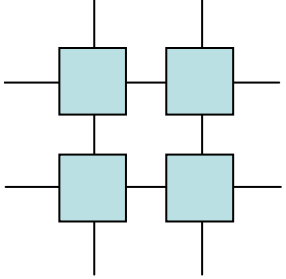
- Traveling Salesman Problem is a classic supercomputing problem
- The chosen algorithm is a typical Producer-Consumer approach
- Is representative for global optimization problems
- May achieve good speedup



Cluster Computing

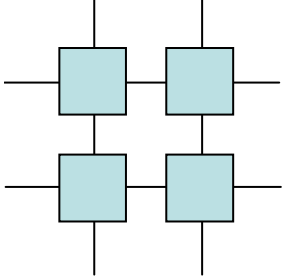
Race Trap





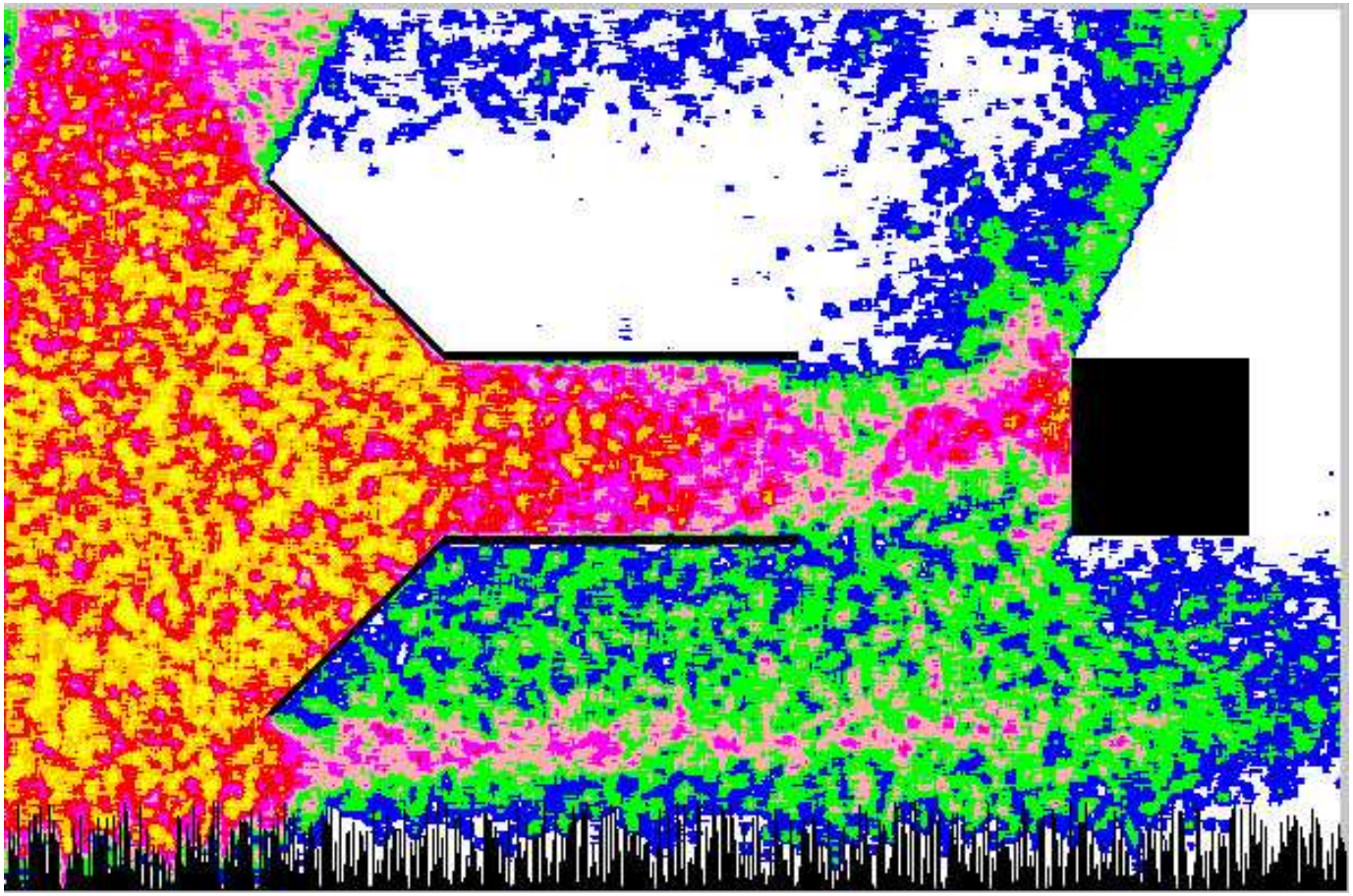
Wind Trap

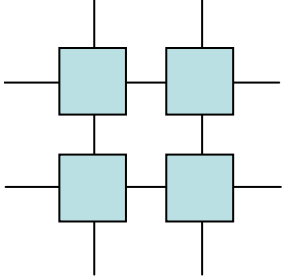
- Virtual wind-tunnel
- An actual scientific application
- 2D – but a 3D model exists that use the same access pattern
- Represents the pipelined application type
- Can achieve really good speedup



Cluster Computing

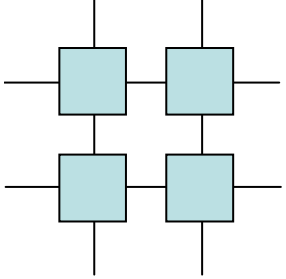
Wind Trap



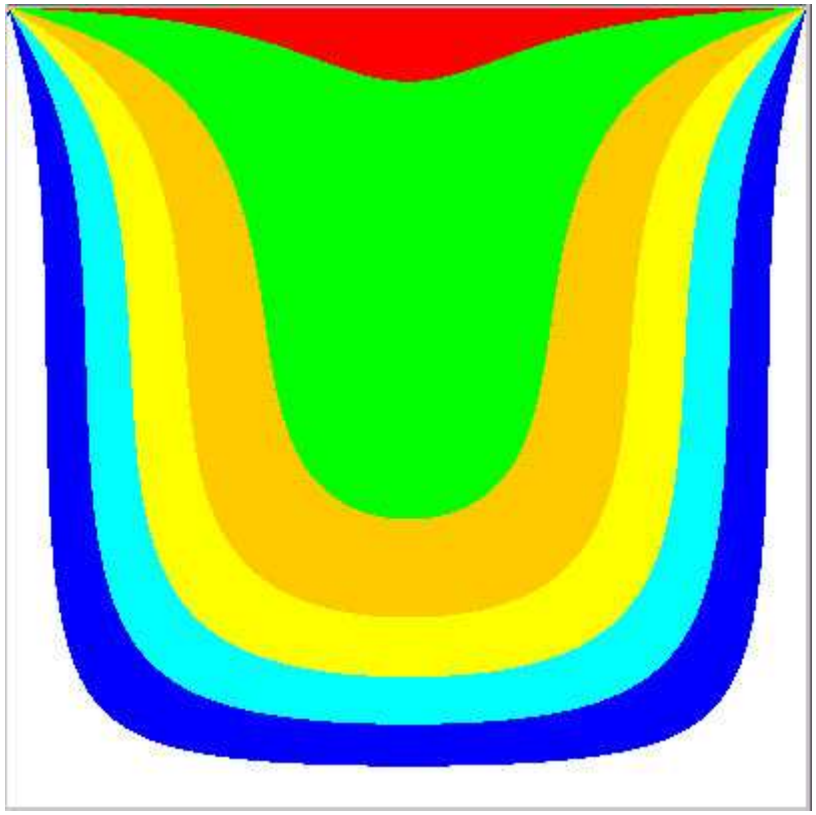


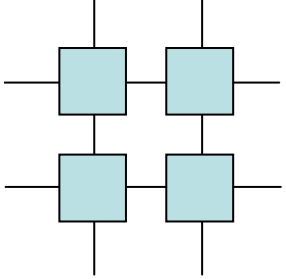
Frosty Trap

- Successive Over Relaxation
- A very common computational kernel in many scientific applications
- A typical example of grid-communication applications
- Can achieve very good speedup



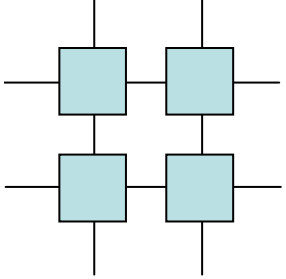
Frosty Trap





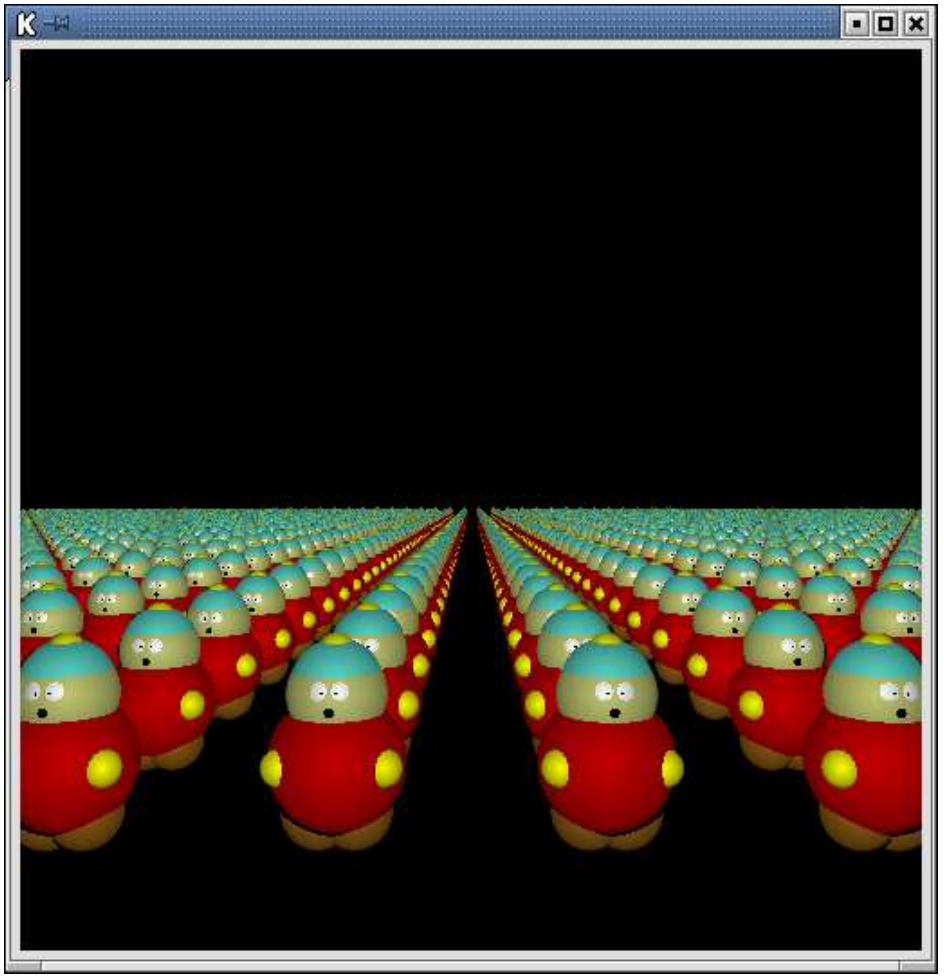
Clone Machine

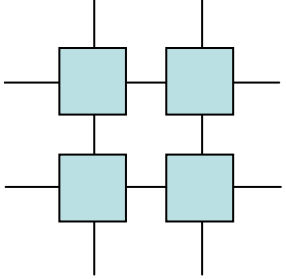
- Ray tracing is a real computational problem
- An example of an application that can achieve perfect speedup with small problems and good speedup on large (real) problems



Cluster Computing

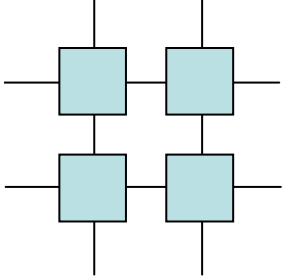
Clone Machine





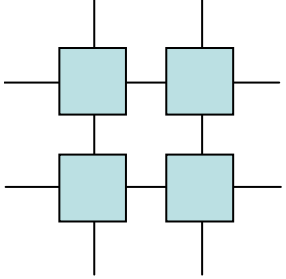
eScience Track

- eScience is a new field at KU
  - Masters of eScience (cand scient escience) will start September 1st
- Starting next year this class will be given in the eScience context
  - Thus there are new assignments on their way
- You can choose to do these assignments instead of the original ones
  - The older ones are well tested the new ones are not!!!



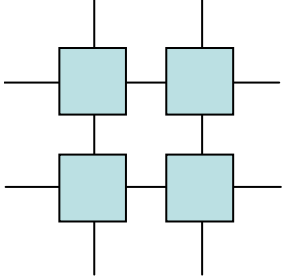
Tumor treatment

- Monte Carlo simulations are examples of applications of the type we call embarrassingly parallel
- A typical example of an compute intensive application with many independent sub-results
- Very simple to write
- Achieves very good speedup

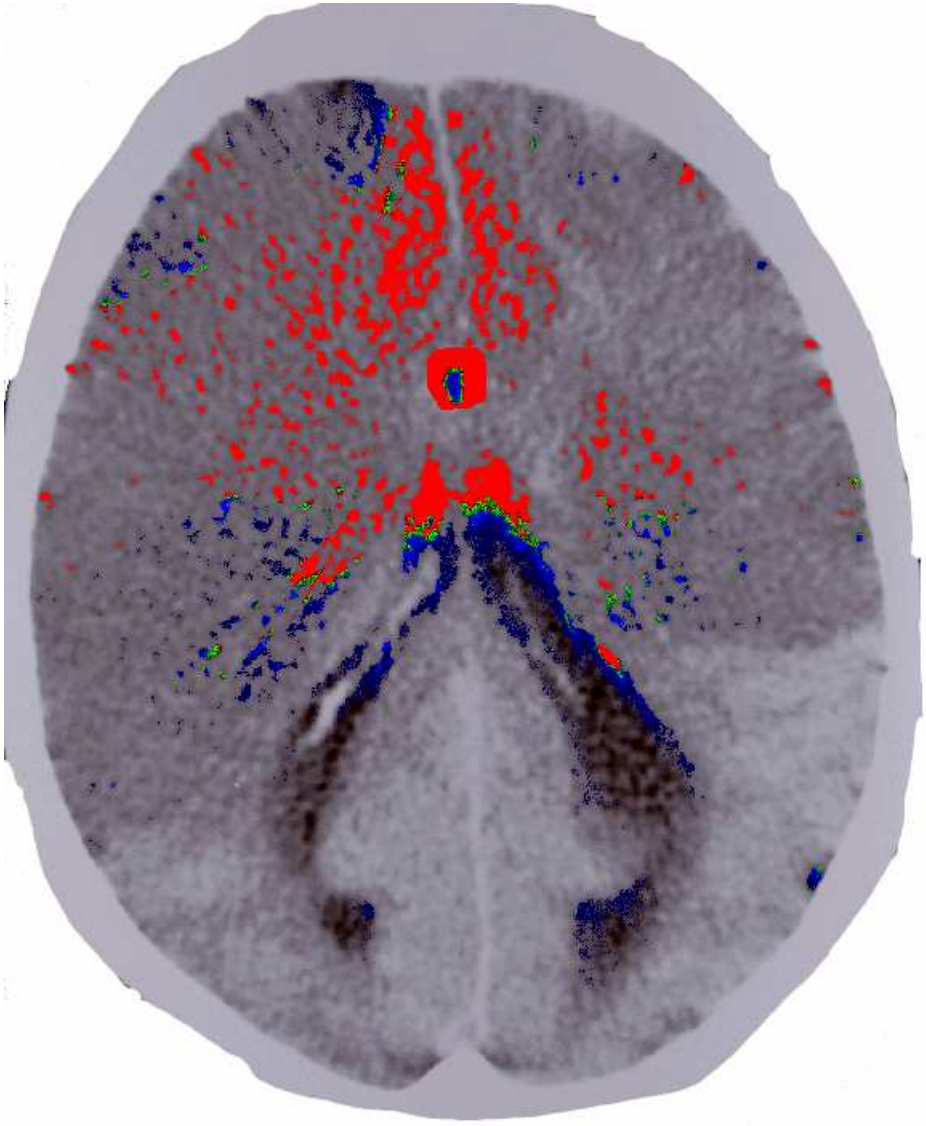


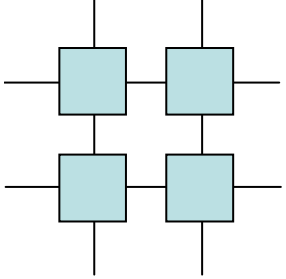
Cluster Computing

Tumor treatment



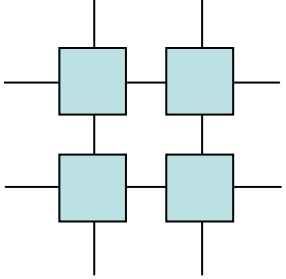
Tumor treatment



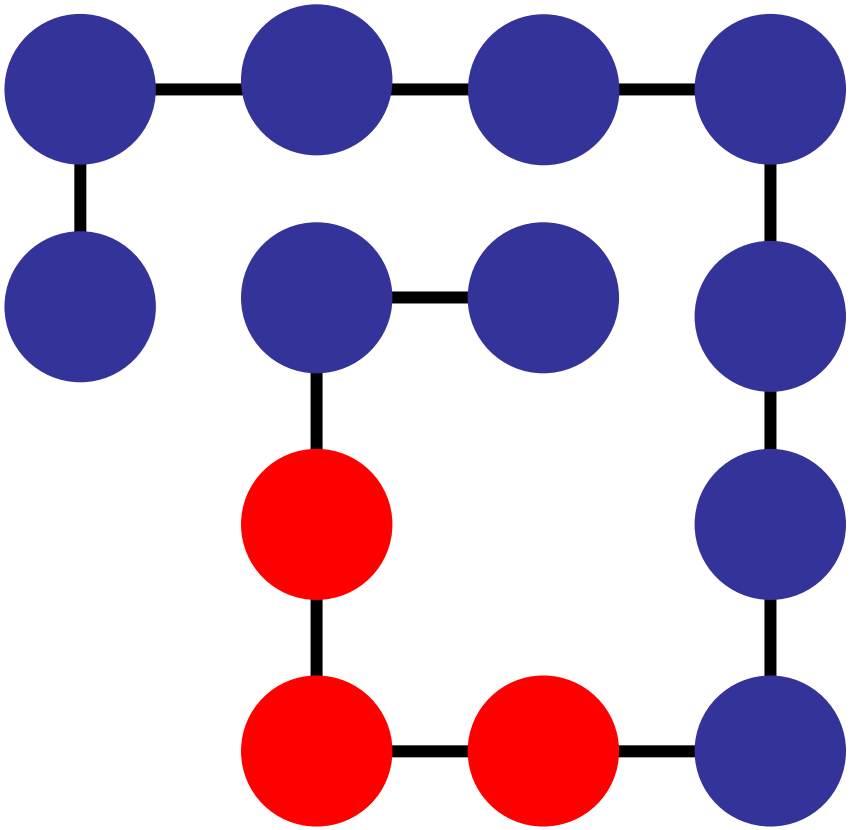


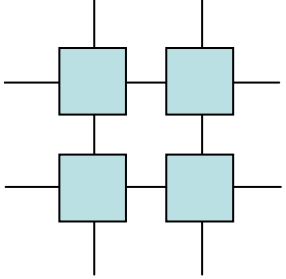
Protein Folding

- Protein is an actual supercomputing problem
- The chosen simplifications and algorithm is a typical Producer-Consumer approach
- Is representative for global optimization problems
- May achieve good speedup



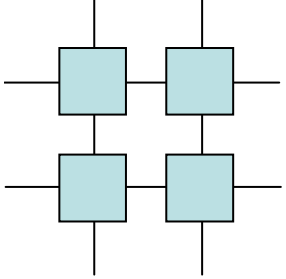
Protein folding





More to come

- Windtrap and frosttrap may not change much
- Clone will probably be replaced with a particle simulation of sorts



Exam

- 3 of your mandatory assignments written into one delivery
  - Plus perhaps a paper only assignment