UiT

THE ARCTIC
UNIVERSITY
OF NORWAY

INF-3201 - Assignment 2 – Shared Memory

Edvard Pedersen
Edvard.Pedersen@uit.no

UNIVERSITY OF TROMSØ

Assignment outline

- Choose a piece of software to parallelize
- Parallelize it with shared-memory techniques
- Evaluate speedup
- Deliverables
  - Report
  - Code

Environment

- Your own computer
- The lab computers
- The cluster
- Use whichever you'd like

- Parallelize with OpenMP (preferred), Pthreads, java.util.concurrent, C++11, Boost and so on
- Again, use whichever you'd like

How to choose the software to use

- The example I use: https://github.com/naftaliharris/markovian
  - A relatively simple chess engine
  - Reasonably simple to find hooks for parallelization
  - Reasonably simple to parallelize
- But this might not be interesting for everyone, therefore, you can choose your own
  - One of your own projects
  - Some piece of software you use
  - Anything at all really
- **But ask me first!**

How to choose the parallelization library

- OpenMP is the «default»
  - Industry standard together with Pthreads
  - Similar to Cilk+
- You can use other approaches, as long as they are shared-memory-focused (so no MPI)
  - They also have to be truly multithreaded (so python isn't quite as easy at it might seem at first

Example programs

- A (very) basic example with the hand-out, which shows some of the OpenMP constructs you can use

Using the OpenMP environment

- Compile with gcc, use the -fopenmp flag
- That's it!

Assessment criteria

- Your solution
  - Does it fulfill the requirements?
  - Is it well-commented and understandable?
- Your report
  - Have you critically evaluated your solution?
  - Have you adequately explained your solution?
  - Have you made your assumptions clear and seperate from your measurements?

Requirements

- Analyze a sequential program
- Parallelize it with shared-memory techniques
- Analyze your solution
- Document your process and results

- Note:
  - Speedup is not a requirement
  - You are not limited to OpenMP or Markovian

Practical details

- Report
  - Short report describing your algorithms and results, as well as reasons for choosing this approach
  - A critical review of your acheived performance (this requires profiling and/or tracing)
  - Accompanying code should be commented
- I am available by e-mail (edvard.pedersen@uit.no) at most hours

Report

- Describe how you have approached the problem (e.g. «profiling shows that functionX() is expensive, so I have paralellized this», «tracing shows that the workload goes from X to Z at this point, which means that the work has to be distributed like so»)
- Try to make assumptions clear, and seperate from measurements
- What I want to know:
  - How did you solve the problems?
  - Why did you solve them in this way?
  - Are the results of your approach any good?

Deadline

Report and code: Thursday October 8th

Resources

- http://openmp.org/
- http://chessprogramming.wikispaces.com/
- http://www.ompp-tool.com
- https://github.com/naftaliharris/markovian