UiT

THE ARCTIC
UNIVERSITY
OF NORWAY

INF-3201 - Assignment 1 - MPI

Edvard Pedersen
Edvard.Pedersen@uit.no

UNIVERSITY OF TROMSØ

Assignment outline

- Parallelize embarrassingly parallel problem with MPI
- Run it on a cluster
- Evaluate speedup
- Deliverables
  - Report
  - Code

Environment

- Uvrocks.cs.uit.no
- 80 nodes, 160 cores
- Log in with UiT username/password
  - If you’re having trouble, let us know!
- Use supplied script to start jobs
- Performance analysis with profiling and tracing tools available

Example programs

- Some sample programs included in hand-out
- Shows basic MPI usage
- Make sure these run successfully before starting your own coding!

Using the OpenMPI environment

- Compile with mpicc
- Run with mpirun
- When you SSH in, use the –X parameter to enable X forwarding
- The run.sh script can be used to easily run your program in a few different settings

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

- Implement two parallel versions of the given Mandelbrot code, using MPI
  - One using a static load balance scheme, the other using a dynamic scheme
  - No shared memory! (Vector instructions are fine)
- The solutions can not use any shortcuts that reduce the functionality of the program
  - They must produce the same CRC
  - All the pixel data has to be sent to the master node
- Evaluate your implementation using profiling and tracing tools
  - Anomalies and lackluster optimizations must be explained in the report

Practical details

- Profiling/tracing tutorial next week
- Best solutions (in terms of time) will be posted, unless you do not want your results published (let me know in your report)
- Report
  - Short report describing your algorithms and results
  - 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?

Hints

- Use the run.sh script as a basis for your testing and experiments
- Don't hog the cluster, use a small number of samples and nodes
- Change the variables (zooms, resolution, number of nodes) when doing experiments, to make sure your solution scales as well as possible
- Use the profiling tools that are available during development (don't just tack it on at the end)
- Use the tracing tools when implementing the MPI stuff, it will enable you to pinpoint communication and load balance problems quickly
- If you get stuck, send me an e-mail

Deadline

Report and code: Thursday September 17th

Resources

- http://www.openmpi.org
- https://sourceware.org/binutils/docs/gprof/
- https://github.com/gperftools/gperftools
- https://github.com/scalability-llnl/ravel

The competition

- After I have reviewed your hand-ins, I will publish the list of the 3 best-performing solutions
- If you do not want to be on the list, send me an e-mail, or put it in your report
- The settings for the test will be published along with the list (resolution, number of zooms, number of hosts etc.)
- Only assignments that pass can make it on the list