Universität Rostock | Fakultät für Informatik und Elektrotechnik
Institut für Informatik, D 18051 Rostock

Scalable Computing (Winter 2010/2011) Laboratory Course Assignments

Parallel Matrix Multiplication Using *phtreads*, *OpenMP*, and *MPI*

Oct 28, 2010

Prof. Dr. Peter Luksch

Fon +49(0)381 498-7561
Fax +49(0)381 498-7522
Mail peter.luksch@uni-rostock.de

http://wwwvhr.informatik.uni-rostock.de/peter.luksch/

Institut für Informatik

Fakultät für Informatik und Elektrotechnik
18051 Rostock

www.informatik.uni-rostock.de

Lehrstuhl für Verteiltes Hochleistungsrechnen (VHR)

Prof. Dr. Peter Luksch
Sitz Albert-Einstein-Straße 21
18059 Rostock
Fon +49(0)381 498-7561
Fax +49(0)381 498-7522

wwwvhr.informatik.uni-rostock.de/

This document describes the assignments of the Laboratory Course over the whole semester.

Administrative Issues

Please set up a time schedule for delivery of the individual milestones. Note that the exam must be completed before March 25, 2011. Also note that I will only fix a date for the oral exam after you have delivered the complete solution to the assignments of the laboratory course. Solutions should be delivered within the lecture period, milestone by milestone. Late submission is possible until Febrary, 27, 23:59.
Submit the solution as an e-mail attachment to your lab course supervisor. The attachment should be a zip archive containing the following:

- documentation that describes the use of the programs
- a design document that describes the structure of the software
- the source code with a makefile
- a number of test cases (matrices) and a script that runs the test cases automatically and records performance data.

All text documents should be delivered in PDF format.

Problem Specification

The multiplication of two matrices $A, B \in \mathbb{R}^{n \times n}$ is to be implemented as

1. a sequential program
2. an OpenMP shared memory program
3. an explicitly threaded program (using the *pthread* standard)
4. a message passing program using the MPI standard
5. a hybrid parallel program, where threads located on the same node communicate using OpenMP and processes on different nodes use message passing

The milestones:

- The **sequential program** will be used to test the correctness of the parallel programs by comparing the results. It also serves as a reference for later measurements of speedup and scalability. This implies a number of requirements.

  In a first step, list these requirements and find out how to meet them. For instance, the program must be able to deal with matrices of arbitraty size. Therefore, the program should dynamically allocate the required amount of memory.

  In the second step, the sequential program has to be implemented, and a number of test runs with matrices of different size are to be executed. Performance Measurements should be carried out. Compare the performance figures of your program to that of your colleagues' programs. Optimize your sequential program if necessary.

- **OpenMP.** Think about different approaches to shared memory parallelization of matrix multiplication and describe them in detail. Implement a selection of variants and analyze their performance.

- **pthreads.** Implement one of the variants that you have selected for OpenMP implementation as an explicitly threaded program, using the pthreads standard. Do the same performance measurements that you have done for the OpenMP program and compare performance data.

- **MPI.** Consider a number of parallelization approaches for distributed memory systems. Implement a selection of them using the MPI standard. Measure and analyse their performance. Compare the performance of the distributed and shared memory parallelizations for a reasonably wide range of numbers of processor cores.
- **Hybrid parallelization with OpnMP and MPI.** Implement a hybrid parallel matrix multiplication that uses OpenMP for intra-node communication, and MPI for inter-node communication. Compare its performance to the pure MPI solution.

Platforms for Program Execution

Shared memory performance should be measured on *atlas* first. After completing the message passing version on the cluster (*sirius*), measure performance of the shared memory implementations on a single node of the VHR cluster (*sirus*). Users from outside the Domain *informatik.uni-rostock.de* can log in to these systems using their Computer Science account via *ekas.informatik.uni-rostock.de*. From your local UNIX system call *slogin -XA atlas.informatik.uni-rostock.de*.
If you have an account in the computer science domain, you will have access to *atlas.informatik.uni-rostock.de*. For an account on *sirius.informatik.uni-rostock.de*, please contact Mr. Bernd Kunde, Bernd.Kunde@uni-rostock.de.