Parallel programming techniques for engineering problems  
(VNr. 6221807, 6221808)  
SS 2014  

Prof. Dr. Markus Uhlmann  

Version March 26, 2014  

1 Time and location  

Time: Wednesday 11:30 – 13 h  
Start date: 16.4.2014  
Location: HS 93, building 10.81  

2 Contact  

Office: Raum 122, Geb. 10.81  
Hours: by appointment  
Phone: 0721-608 47245  
Email: markus.uhlmann@kit.edu  

3 Scope of this course  

- To convey the basic knowledge about parallel computing possibilities and its limitations.  
- To enable the students to analyze a given problem from CFD (and beyond) and assess the potential for an efficient solution using parallel computing techniques.  
- To transmit the scope, syntax and practical application of the message passing paradigm, using the standard ”MPI”.  

4 Course material  

Will be made available under ILIAS. Please susbscribe to this course under the following URL:  
https://ilias.studium.kit.edu/gotoProduktiv_crs_307543.html  

5 Prerequisites  

- successful participation in Module “Advanced Fluid Mechanics” (PM1) or equivalent  
- programming skills in either Fortran, C or C++  
- if possible, please bring your own laptop  

6 Exam  

Written exam with a duration of 60 minutes (date to be announced).
7 Planning and content of the course

Lecture 1 (16.4.): General introduction to parallel programming
Background on hardware; software paradigms; measuring efficiency; network topologies.

Lecture 2 (23.4.): General introduction to MPI
“hello world!”

Lecture 3 (30.4.): MPI point-to-point communication
“send/recv”

Lecture 4 (7.5.): Case study – parallel search problem
“search”

Lecture 5 (14.5.): MPI collective communication
“pi”

Lecture 6 (21.5.): Case study – 2D Poisson solver
“Jacobi”

Lecture 7 (28.5.): Non-contiguous data & mixed datatypes
“search”

Lecture 8 (4.6.): Virtual topologies & Communication subsets
“search”

Lecture 9 (18.6.): Use of linear algebra libraries – dense linear system solver
“scaex”

Lecture 10 (25.6.): Some examples of parallel applications – Navier-Stokes solvers
Parallel wavelet transform; spectral methods for DNS of single-phase flow; finite-difference method for particulate flow DNS.

Dates for additional exercises: 2.7., 9.7., 16.7.
Solution of hands-on exercises.

8 Further Resources

- NCSA online courses on parallel programming and MPI:
  http://www.citutor.org/users/index.php

- A complete reference of the MPI library standard is available at NETLIB:

- A useful short summary of the syntax and use of each MPI command can be accessed at the following URL:
  http://www-turbul.ifh.uni-karlsruhe.de/uhlmann/mpi2/www/index.html

- The user guides for SCALAPACK and BLACS are also available at NETLIB:
  http://www.netlib.org/scalapack

References


In practice, complex designs often include complicated tradeoffs between sequential architectures and parallel architectures, in order to achieve the best overall design. In Vivado R HLS, these tradeoffs are largely controlled by the user, through various tool options and code annotations, such as #pragma directive. The chapter focuses on a limited subset of optimization techniques. And the application complexity generally increases in the later chapters. We start with a relatively simple to understand finite impulse response (FIR) filter in Chapter 2 and move on to implement complete video processing systems in Chapter 9.

Parallel Algorithm - Design Techniques - Selecting a proper designing technique for a parallel algorithm is the most difficult and important task. Most of the parallel programming problems may have more than one solution. In this chapter, we will discuss the following designing techniques for parallel algorithms: Divide and conquer. Greedy Method. Dynamic Programming. Backtracking. Branch & Bound. Linear Programming. Divide and Conquer Method. In the divide and conquer approach, the problem is divided into several small sub-problems. Then the sub-problems are solved recursively and combined to get the solution of the original problem.

Grand Challenge Problems. Slide 3. A grand challenge problem is one that cannot be solved in a reasonable amount of time with today’s computers. Obviously, an execution time of 10 years is always unreasonable. Examples: Modeling large DNA structures, Global weather forecasting, Modeling motion of astronomical bodies.