



Technical University of Denmark



High Performance Computing

FORTRAN, OpenMP and MPI

41391

Content of Course

- Week 1: FORTRAN



Prof. J. H. Walther
DTU MEK
jhw@mek.dtu.dk

- Week 2: OpenMP
- With 02614



Asso. Prof. B. Dammann
DTU Compute
beda@dtu.dk

- Week 3: MPI



Prof. J. H. Walther
DTU MEK
jhw@mek.dtu.dk

Learning Objectives

- read programs written in FORTRAN.
- write programs in FORTRAN.
- read programs with OpenMP directives.
- write programs with OpenMP directives.
- read programs using MPI.
- write programs using MPI.
- understand the difference between shared and distributed memory parallelism.
- perform serial benchmarking of code.
- perform code debugging.
- measure parallel efficiency
- use Amdah's law.

FORTRAN

High level procedural language developed by IBM in the 1950ies:

- Mathematical **FOR**mula **TRAN**slating System.
- Suited for scientific and engineering computing.
- ANSI standards: '66, '77, '90, '95, '03, '08, '18.
- Key features:
 - '77: arrays, characters, **modular** programming.
 - '9x: allocatable arrays, **object-based** programming.
 - '0x: **object-oriented**, generic programming.

OpenMP

- **Open Multi-Processing (OpenMP)** is a standard **Application Programming Interface (API)** that supports multi-platform **shared memory** multiprocessing programming in C/C++ and FORTRAN.
- OpenMP consists of compiler directives, library routines, and environment variables that enable shared memory parallelism and execution.
- Relatively easy parallelization:
 - Line-by-line strategy for distribution of work load.
 - No need for distributing the data/memory!
- Limited scalability (beyond 4-8 cores):
 - Congestion of the shared memory network.
 - Expensive hardware – limited memory.

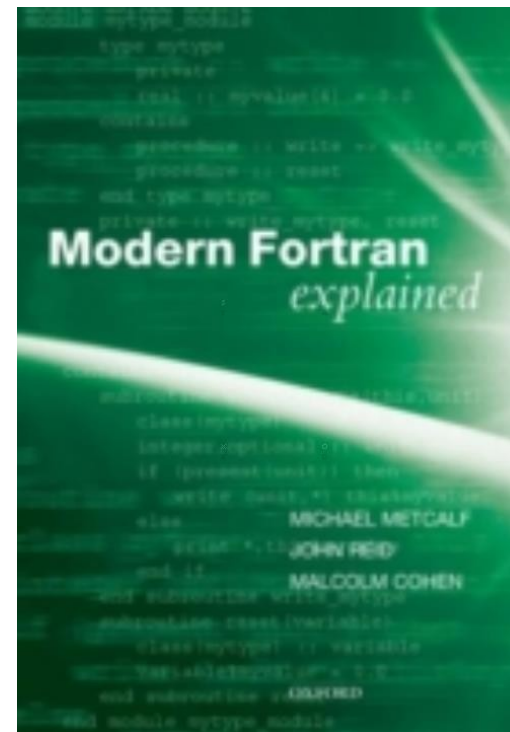
MPI

- **M**essage **P**assing **I**nterface (**MPI**) is the de-facto standard for programming portable message-passing parallel applications on **networked** computers (clusters).
- Has bindings to C/C++ and FORTRAN and is available from single core laptops to massively parallel supercomputers.
- Relatively difficult to implement:
 - Explicit spatial distribution of the problem: data/memory.
 - Complete revision of the algorithms and implementation may be required.
- Good scalability (to thousands/million of cores):
 - The memory is local – so access to local memory is efficient.
 - Inexpensive hardware – unlimited memory/processors.

Content FORTRAN 95

- Day 1:
 - Introduction.
 - Language elements.
 - Expression and assignments.
 - Control constructs.
- Day 2:
 - Program units and procedures.
- Day 3:
 - Array Features.
 - Makefiles.
- Day 4:
 - Specification statements.
 - Intrinsic procedures.
- Day 5:
 - Data transfer.
 - Operations on external files.
 - Advice for the road.
 - Fortran 2003/2008 extensions.

Class Homepage:
learn.inside.dtu.dk



Modern Fortran Explained
John Reid, Malcolm Cohen & Michael Metcalf

Computer accounts

- Computer system: login.gbar.dtu.dk:
 - UNIX/linux (SunOS/linux):
 - www.gbar.dtu.dk
 - www.gbar.dtu.dk/faq/43-thinlinc (remote access - Windows).
 - Access:
 - Use DTU username/passwd (guest account for non-DTU students).
 - Remote access using Thinlinc or ssh:
 - ssh -Y s111111@login.gbar.dtu.dk
 - continue to linux nodes by typing the command: `linuxsh -X`
 - UNIX/linux documentation available in **man** pages. Accessed in a shell window/terminal through the command 'man <command>' e.g.,
man ls

Compiler

- Sun FORTRAN 90/95 compiler:
 - To setup the ENV variables to point to the Sun compiler on the gbar, type, in the terminal:
 - `module add studio`
 - `module add mpi/3.1.3-oracle-12u6`
 - You may also download the compiler at:
www.oracle.com/technetwork/server-storage/solarisstudio/downloads/index.html
 - Documentation: type '`man f90`' or '`f90 -help`' in the terminal.
- To compile a simple serial program:
`f90 -free example.f`
- To compile a simple parallel program:
`mpif90 -free example.f`

Lectures and Exercises

- Lectures: 9:00 – 11:30 on zoom.
- Exercises: 13:00 – 16:00 on zoom.
 - Work in groups of 1-2.
 - Hand in report (minimum 4 out of 5) at the end of each week; report = source code (+ plots/figures)