

750711, Parallel Programming Languages

3 hours per week, 3 credit hours, prerequisite: **none**

Teaching Method:

30 hours Lectures (2 hours per week), 7 hours Seminars (1 per 2 weeks), 8 hours

Laboratories (1 per 2 weeks)

Aims:

This module aims to cover a variety of paradigms and languages for programming parallel computers. Topics covered in depth include parallel programming techniques with their implementations in a parallel environment; shared-memory and message-passing models, process synchronization and data sharing, communication; converting sequential algorithms into equivalent parallel algorithms; improving performance of parallel algorithms. Several tools for debugging and measuring the performance of parallel programs will be introduced and some example languages will be covered.

Learning Outcomes:

On completion of this module, the student should be able to:

- Know a variety of paradigms and languages for programming parallel computers.
- Know how to convert sequential programs for single processor computers into faster working programs that give the same results when run on parallel computers, especially those with globally shared address spaces.
- Understand the fundamental aspects of parallel processing.
- Implement the new concepts of programming languages in parallel programs.
- Be familiar with performance measures of parallel programs and the fundamental limits on their speedup.

- Comprehend the distinction between different programming models.
- Understand the theoretical limitations of parallel computing such as intractability.
- Design and analyze simple parallel algorithms.
- Write efficient parallel application programs.
- Write programs using thread programming methods for a shared memory computer model.

Textbooks and Supporting Materials:

- 1- Barry Wilkinson, Michael Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2ed, Pearson/Prentice Hall, 2005. ISBN 0-13-140563.
- 2- Salim G. AKI, Parallel Computation Models and Methods, Prentice Hall, 1997.
- 3- Michael J. Quinn. Parallel Programming in C with MPI and OpenMP, McGraw Hill (2004), ISBN 0-07-282256-2
- 4- Shirley A. Williams, Programming Models for Parallel Systems, John Wiley, 1990
- 5- P. I. Fleming (editor), Parallel Processing in Control: the transputer and other architectures, Peter Peregrines ltd, 1988.
- 6- Joel M. Crichlow, An Introduction to Distributed and Parallel Computing, 2nd edition, Prentice Hall, 1997.
- 7- Michael J. Quinn, Designing Efficient Algorithms for Parallel Computers, McGraw Hill, 1987.
- 8- Foster, I. T., Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, (ISBN 0-201-575-949), Addison-Wesley 1995. (Very good on message-passing style of programming)
- 9- M. L. Scott, Programming Languages Pragmatics, Morgan Kaufman Publishers, 2000
- 10- M. E. C. Hull, D. Crookes, D. J. Sweeney (editors), Parallel Processing: the Transputer and its Applications, Addison Wesley, 1994
- 11- Culler, D. E. and Singh, J.P., with Gupta A., Parallel Computer Architecture: a hardware/software

- approach, (ISBN 1-55860-343-3), Morgan Kaufmann 1999. (Very good book on architecture level)
- 12- Hockney, R. W. and Jesshope, C.R., Parallel Computers 2, (ISBN 0-862-748-124), Adam Hilger 1988. (A good, albeit old-fashioned reference text for fundamental techniques)
- 13- David E. Culler and Jaswinder Pal Singh, with Anoop Gupta. Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann, 1998. ISBN: 1-55860-343-3.
- 14- George Almasi and Allan Gottlieb, Highly-Parallel Computing, 2nd Edition, Benjamin-Cummings, 1994.
- 15- Ted G. Lewis, Hesham El-Remini, Introduction to Parallel Computing, Prentice Hall 1992.
- 16- Grama, A. Gupta, G. Karypis and V. Kumar. Introduction to Parallel Computing (2nd edition), Addison Wesley (2002), ISBN 0-201-64865-2.
- 17- H. El-Rewini and T. G. Lewis. Distributed and Parallel Computing, Manning (1997), ISBN 0-13-795592-8.
- 18- R. J. Barlow and A. R. Barnett, Computing for Scientists: Principles of Programming with FORTRAN 90 and C++, John Wiley and Sons, 1998.
- 19- Andrei Alexandrescu, Modern C++ Design: Generic Programming and Design Patterns Applied, Addison-Wesley, 2001
- 20- Robert Robson, Using the STL: The C++ Standard Template Library, Springer, 1997.
- 21- Gregory V. Wilson and Paul Lu (editors), Parallel Programming using C++, MIT Press, 1996.
- 22- Thuan Q. Pham and Pankaj K. Garg, Multithreaded Programming with Windows NT, Prentice Hall, 1995.
- 23- Gregory V. Wilson, Practical Parallel Programming analysed, MIT Press, 1995.
- 24- Foundations of Multithreaded, Parallel, and Distributed Programming, Addison-Wesley, 2000
- 25- E. V. Krishnamurthy, Parallel Processing Principles and Practice, Addison Wesley, 1989

*** Plus some Research Papers on the topics**

Synopsis:

- 1- An Introduction to parallel programming languages: specifying parallel processes;
- 2- Categories of parallel programming models: sequential processing, Array processing, pipeline processing, shared memory processing, message passing, Data Parallel programming, functional programming, logic programming, object-oriented programming;
- 3- Software architectures: Semaphores, Monitors, Remote Method Invocation, the Ada rendezvous, message passing semantics;
- 4- Multithreaded programming: Threads, Synchronization techniques, Java threading model, Applications
- 5- Multiple process programming: Sockets, Messages, Applications, Client/Server models
- 6- Multiple process programming: CORBA
- 7- RMI: Basic principles, Techniques for using
- 8- MPI: Basic message, Synchronization, first MPI program, Scatter/gather, Broadcast messages, Groups/contexts, I/O
- 9- Parallel programming languages and algorithms: parallel language and algorithm design for the array processor: Actus; Von Neumann-type languages: Concurrent Pascal, Communicating Sequential Processes (CSP) and Occam, Distributed Processes (DP), Ada, Linda, C++, Java; Non-Von Neumann-type languages: functional programming (FP), Lisp.
- 10- The transputer implementation of Occam; Control application of Transputers
- 11- Detection of parallelism within expressions, parallelism in Tree structures, determining parallelism between blocks of programs,
- 12- Restructuring for Parallel Performance: Parallelising Compilers; Loop Transformations; Data Transformations; Dependence Compiler Strategies; Analysis; Reducing Parallelism; Parallelizing serial programs
- 13- Examples of Parallel Algorithms: Sorting and searching algorithms; Cyclic Reduction; Iterative Algorithms (Jacobi, Gauss-Seidel and Red-Black Orderings); Divide-and-Conquer Algorithms

Assessment: Two 1-hour mid-term exams (10% each); Assignments (10%) (two assignments on parallel models of computation); writing a research paper / presentation / final project (20%); 2-hours Final Exam (50%).