# An M.Sc. Level Course of Studies in Parallelism

**Alejandro Teruel[1], Marianela Lentini[2], Carlos Figueira[1]**
**Universidad Simón Bolívar, Apartado 89000**
**Caracas 1080A, VENEZUELA**

**Abstract**
Two years ago, at the Universidad Simón Bolívar, we designed and successfully implemented an ongoing set of M.Sc. level courses in Parallelism to help overcome the problems besetting graduate students who wish to specialize in the area and tackle problems such as choosing a parallel programming hardware and software platform, developing parallel programs for it and building the necessary tools to support parallel software development.

The purpose of this paper is to present the current status of the Universidad Simón Bolívar's course of studies in Parallelism.

**Keywords:** Parallelism, education, softwaredevelopment.

## Introduction

Choosing a parallel programming hardware and software platform, developing parallel programs for it and building the necessary tools to support parallel software development all require a thorough knowledge and understanding of many diverse and interleaved issues in parallel hardware architecture, parallel algorithm design, tools for parallelism and the theory of parallel processes. A formal course of studies in such topics is becoming necessary in order to provide a coherent structure to what otherwise becomes a scattered, haphazard, repetitive, superficial or, at best, lopsided coverage of topics. Two years ago, at the Universidad Simón Bolívar (USB), we designed and successfully implemented an ongoing set of M.Sc. level courses in Computer Science to help overcome such problems. The purpose of this paper is to present the current status of the USB's course of studies in Parallelism.

## Objectives

The current course of studies emphasizes the MIMD distributed memory model of parallelism. This emphasis is due mainly to the local demand for our M.Sc. graduates and to the hardware acquisition restrictions imposed by our extremely small budget. However, the course stucture is independent of this bias and its contents can easily be

[1] Department of Computer and Information Technology
[2] Department of Mathematics

focused towards other models.

The objectives of the course of studies are:

- To provide a coherent coverage of the field of parallelism;
- To prepare students for the wide variety of roles they will have to play in the near future as regards the selection, development, and use of parallel technology.

Obtaining the fastest possible running parallel programs is the raison d'etre in the field of Parallelism today, hence the disproportionate intertwining of programming techniques, performance analysis and software and hardware platform interdepencies that must be taken into consideration during parallel software development. While we have found it convenient to factor out topics into the more conventional categories of hardware (Parallel Architectures), system software and tools (Parallel Tools) and application software (Parallel Algorithms, Parallel Numerical Methods), each course can be better described as emphasizing one category's point of view over the others rather than narrowing down on the study of any one category to the exclusion of the others --the latter simply misses the nature of the whole field.

**Requirements**

The course of studies in parallelism presupposes basic knowledge of computer architecture, concurrency, operating systems, algorithm design and graph theory, as provided in a typical undergraduate curriculum in Computer Science. Students with a major in related fields such as Mathematics and Electronic Engineering have also been able to successfully complete the course of studies.

**Courses**

The course of studies consists of the following courses:

- Parallel Architectures;
- Parallel Algorithms;
- Parallel Numerical Methods I;
- Parallel Numerical Methods 2;
- Tools for Parallelism;
- Theory of Parallel Processes.

In order to satisfy the M.Sc. degree requirements in Computer Science at the USB, candidates must successfully complete at least eight Computer Science courses and an M.Sc. thesis. The course work must include the following kernel:

- Mathematics for Computer Science;
- Algorithm Analysis;

- Theory of Computation.

The planned duration of studies leading up to the degree is four twelve-week terms; this spans a calendar year and a quarter.

A more detailed description of each course follows.

**ParallelArchitectures**

The objective of this course is to provide a programmer of parallel machines with the knowledge necessary to:

- Choose the best parallel architecture model for a given problem;
- Understand the limitations of a specific parallel machine or architecture;
- Tune programs taking into account the features of a specific parallel architecture.

Course outline

The course looks at ways of speeding up the key subsystems in order to prevent their becoming bottlenecks on parallel machines. The impact of such architectural decisions on cost, speed, scaleability and tool and programming complexity is also carefully considered. A highly recommended form of drawing all the threads of the course together at its end is setting a final assignment which consists of selecting a parallel machine for a (fictitious) client company operating in a specific industrial sector (e.g. oil exploration and extraction or prothesis design) and using order of magnitude restrictions on the budgeted cost of the system. Ease of operation, use and programming as well as software availability, functionality and performance must also be taken into account, together with more specific hardware and supplier characteristics.

Topics

- Introduction.

  Why parallelism? Parallelism vs. distributed systems. Flynn's taxonomy of parallel architectures. Applications of parallelism.
- Measuring performance on parallel architectures.

  Throughput and latency. Speedup and Amdahl's Law. Peak and sustained performance. MIPS, MFLOPS and workloads. Using kernels and benchmarks.
- Pipelining and Vector machines.

  Pipelines and their hazards. Superscalar machines. Vector registers and vector instructions. Stride. Interleaved memories and memory conflicts. Chaining. Vector masks. Hockney's metrics: $n_v$, $n_{1/2}$, $r_\infty$. Cache and vector machines.
- MIMD machines.

  Shared memory vs. distributed memory. Interconnection networks for shared

memory machines: Clos networks vs. Banyan networks. Synchronization methods. Cache on multiprocessor systems. Interconnection networks for distributed memory machines: fixed topology vs. reconfigurable topologies. Cut-through routing, wormhole routing. Input/output subsystems: Disk arrays.

- SIMD machines and massive parallelism.
- Other architectures.

  Systolic architectures. Dataflow machines.
- Case studies.

  Varies from year to year. In 1991, case studies included Intel i860 and INMOS transputer platforms, the IBM 3090VF, nCube and Maspar parallel machines.

References

No textbook covers all the required topics with the necessary level of detail. The following set of references complement each other and are listed in roughly descending order of use:


H. Stone: "High-Performance Computer Architecture". 2° Edition, Addison-Wesley, 1990.

B. Wilkinson: "Computer Architecture: Design and Performance". Prentice-Hall, 1991.

D. Patterson, J. Hennessy: "Computer Architecture: A Quantitative Approach". Morgan-Kauffman, 1990.

G. Almasi, A. Gottlieb: "Highly Parallel Computing". Benjamin/Cummings, 1989

K. Hwang, F. Briggs: "Computer Architecture and Parallel Processing". McGraw-Hill, 1984.

D. Bertsekas, J. Tsitsiklis: Parallel and Distributed Computation: Numerical Methods". Prentice-Hall, 1989.

R. Hockney, C. Jesshope: "Parallel Computers 2". Adam Hilger, 1988.


Some topics require more specific treatment. A complete list of references for the course is available from the authors.

**Parallel Algorithms**

The emphasis of the course is on the design and implementation of efficient parallel algorithms. In general, first algorithms for PRAM models are developed and compared and then their adaptation to actual MIMD machine with distributed memory is studied, attempted and compared to known solutions, where they exist. Programming some of the algorithms is an indispensable part of the course, since it allows the student to grapple with the characteristic difficulties of the area e.g. load balancing, efficient communications and steering clear of deadlocks.

Topics

- Introduction

  The PRAM models (EREW, CREW, CRCW). Distributed memory MIMD

architectures. Measuring performance: speedup, efficiency and cost. A parallel programming language and platform (used for programming assignments e.g. INMOS Parallel C on a transputer based system).

- Parallel searching, sorting and merging.
  Searching. Prefixed sums. Selection. Merging. Sorting.
- Parallel graph algorithms.
  Basic matrix operations. Finding the connected components. All pairs shortest paths. Topological sort. Minimum spanning tree. Eulerian circuits. Travelling salesman and the simulated annealing heuristic.
- The class of inherently sequential algorithms.

The rest of the topics covered are chosen from the following list, time permitting:

- Data structures and parallelism: hashing, heaps, priority queues.
- More graph algorithms: maximal matching, graph colouring, network flows.
- Pattern matching.
- Alpha beta tree search.
- Computational geometry.
- Optimization.

References

The basic textbook for the course is:

S. Akl: "The Design and Analysis of Parallel Algorithms". Prentice-Hall, 1989.

This needs to be complemented by a book which deals with some of the nasty surprises that can crop up for the unwary programmer:

G. Fox , M. Johnson, G. Lyzenga, J. Salmon, D. Walker: "Solving Problems on Concurrent Processors". Volume 1. Prentice-Hall, 1988.

Additional material can be found in:

D. Bertsekas , J. Tsitsiklis: Parallel and Distributed Computation: Numerical Methods". Prentice-Hall, 1989.

S. Bokhari: "Multiprocessing the Sieve of Eratosthenes". Computer, April 1987.

A. Rytter, W. Gibbons: "Efficient Parallel Algorithms". Cambridge University Press, 1988.

Y. Won. S. Sahni: "A Balanced Bin Sort for Hypercube Multicomputers". Journal of Supercomputing, 2 (1988).

Y. Won, S. Sahni: "Hypercube to Host Sorting". Journal of Supercomputing, 3 (1989).

## Parallel Numerical Methods 1,2

These two courses provide a thorough grounding in the state of art in the development of numerical algorithms for parallel architectures, with emphasis on MIMD distributed memory machines. A number of alternatives to each problem are considered; each algorithm is analyzed for efficiency, stability and consistency.

Topics

- Review of parallel architecture and other basic concepts in parallelism;
- Common communication requirements of numerical algorithms;
- Parallel implementation of BLAS routines.
- Direct solution of linear equations. Dense and sparse problems.
- Iterative solution of linear and nonlinear systems.
- Unconstrained optimization. Decomposition of optimization problems.
- Asynchronous iterative algorithms.

References

The main references for the course are:

D. Bertsekas, J. Tsitsiklis: Parallel and Distributed Computation: Numerical Methods". Prentice-Hall, 1989.

J. Modi: "Parallel Algorithms and Matrix Computation". Clarendon Press, 1988.

M. Lentini: "Introducción a los Algoritmos Numéricos en Paralelo". Segunda Escuela Venezolana de Matemáticas, 1989.

M. Ortega: "Introduction to Parallel and Vector Solution of Linear Systems". Plenum, 1988.

Additional material includes:

Y. Saad, M. Schultz: "Data Communication in Parallel Architectures". Yale University Research Report. YALEU/DCS/RR461. 1986.

G. Geist, C. Romine: "LU Factorization Algorithms on Distributed Memory Multiprocessor Architectures". Oak Ridge National Laboratory Report ORNL/TM-10383. 1987.

M. Heath, C. Romine: "Parallel Solution of Triangular Systems on Distributed Memory Multiprocessors". SIAM J. Sci. Stat. Comp. 9 (1988).

C. Ashcroft, S. Eisenstat, J. Liu , H. Sherman: "A Comparison of Three Column-Base Distributed Sparse Factorization Schemes". Yale University Research Report. YALEU/DCS/RR 810, 1990.

A. George, J. Liu, E. Ng: "Communication Results for Parallel Sparse Cholesky Factorization on a Hypercube". Parallel Computing 10 (1989).

M. Heath, E. Ng, B. Peyton: "Parallel Algorithms for Sparse Linear Equations". SIAM Review 33(1991).

## Tools for Parallelism

The objective of the course is to study the kind of tools a parallel software developer has available to him and how much of the parallelizing effort he can hope to have the tools do

for him.

The course has a seminar format. The instructor updates a rather large number of published papers which is distributed amongst the participants for class presentation and discussion. Such seminar classes should be complemented wherever possible by actual hands on experience with representative tools.

Topics

- Vectorizing and parallelizing compilers.
- The mapping problem and granularity.
- Basic communication libraries for distributed memory systems.
- Debugging parallel programs for correctness and for performance.
- Putting it all together: Parallel programming environments

References

Due to the course's seminar structure, the list of readings is constantly changing. The current set of kernel references is:

H. Zima, B. Chapman: "Supercompilers for Parallel and Vector Machines". Addison-Wesley, 1991.

M. Wolfe: Optimizing Supercompilers for Supercomputers". The MIT Press, 1990.

L. Jamieson, D. Gannon, R. Douglass: "The Characteristics of Parallel Algorithms". MIT Press, 1987.

D. Reed, R. Fujimoto: "Multicomputer Networks: Message-Based Parallel Processing". MIT Press, 1987.

C. McDowell, D. Helmbold: "Debugging Concurrent Programs". ACM Computing Surveys. Vol.21 N°4. December 1989.

M. Reilly: "A Performance Monitor for Parallel Programs". Academic Press, 1990.

G. Fox, M. Johnson, G. Lyzenga, J. Salmon, D. Walker: "Solving Problems on Concurrent Processors: Vol. 1". Prentice-Hall, 1988. Chap. 14: CrOS; Chap. 15: CUBIX;

An anthology of twenty to thirty articles is being compiled for the 1992 course. Due to space restrictions they will not be cited in this paper. The interested reader may write directly to the authors for details.

## Theory of Parallel Processes

Efficiently executing a parallel program that produces the wrong results or deadlocks is utterly useless. It is important to understand the underlying theory of parallel processes in order to be able to reason about parallel programs in order to prove their correctness or to show whether certain properties like absence of deadlock, livelock or critical races hold. The key to the conceptual basis for reasoning about parallelism lies in the relationship between concurrency and non-determinism. The course covers two of the many theories explicitly developed to understanding such questions and shows how these theories can

be used to design and reason about parallel algorithms and languages.

Topics

- Introduction.

  The need for a formal theory of parallel processes. Mathematically, what is a process? Varieties of models and theories. Concurrency and non-determinism as key issues.

- The theory of UNITY.

  A logic for parallelism. Architectures and mappings. Program compositions. Examples and applications.

- The theory of CSP.

  Events and processes. CSP operators. CSP's models: traces, failures, divergences. Superimposed obligations. Examples and applications. CSP and Occam.

References

C. Hoare: "Communicating Sequential Processes". Prentice-Hall, 1985.

K. Misra, J. Chandy: "Parallel Program Design: A Foundation". Addison-Wesley, 1988.

## Experience and conclusions

Our experience in setting up and tuning a course of studies in Parallelism has been extremely interesting and very rewarding, in that it has forced us to attempt to integrate and clarify a great deal of scattered material and subtle interplays of software and hardware. We would like to enrich the actual parallel environment so as to provide more hands-on experience and strengthen the coverage of performance modelling techniques, especially as regards statistical modelling techniques. While it is clearly necessary to provide in-depth coverage of representative theories on parallelism in order to appreciate their strengths, weaknesses and subtleties, restricting such a study to two such theories may seem unduly constrictive --in particular, omitting any reference to Petri Net based theories is definitely worrisome. In fact, striking the right balance between depth of treatment and breadth of coverage is a recurring thread throughout the evolution of all our courses.

Student and industry feedback has been very positive, especially as regards the Venezuelan oil industry whose keen interest in the venture has been a constant spur to our efforts. In particular, we would like to thank Alfonso Reinoza, Jorge Baralt, Oscar Meza and Cristina Zoltán of the USB, Raju Karia, Benjamín Sagalovsky, Saúl Buitrago and Enrique Rodríguez of INTEVEP, Rubén Amaya of MARAVEN, Walter Cunto, Kley Visentín and Jorge Gonçalves of IBM Venezuela, Jorge Vidart of ESLAI, Euripides Montagne of the Universidad Central de Venezuela and, of course, all our students, whose comments and enthusiasm have helped shape our programme.