# Experiencias en la enseñanza de Programación Paralela

*Ing. A.De Giusti[1] , Lic. M. Naiouf[2]*

*Laboratorio de Investigación y Desarrollo en Informática[3]*
*Departamento de Informática - Facultad de Ciencias Exactas*
*Universidad Nacional de La Plata*

## Resumen

Se discute una primera implementación de un curso de Programación Paralela como asignatura optativa de quinto año de la Licenciatura en Informática de la UNLP.

Las dificultades más importantes al estructurar este curso en 1994 estaban centradas en la necesidad de definir un contenido teórico (prácticamente con muy pocas referencias por lo novedoso del tema en el ámbito universitario), y sobre todo en estructurar una práctica coherente con los temas teóricos, dentro de la escasa disponibilidad de recursos.

En este trabajo se resume la evolución de los contenidos teóricos y los diferentes modelos de arquitectura sobre los que han efectuado experiencias los alumnos (procesadores heterogéneos distribuidos, transputers, emuladores de DSP). Por otra parte, se plantean algunas de las herramientas de especificación de procesos utilizadas (Redes de Petri extendidas, CSP).

Por último, se analiza el contexto de los conocimientos previos requeridos por el alumno, se mencionan una serie de "problemas tipo" que enmarcan el desarrollo conceptual del curso, y se explicita la transformación a realizar al disponerse desde 1997 del Laboratorio de Cómputo Paralelo en el Departamento de Informática de la UNLP.

**[1] Inv. Principal CONICET. Profesor Tit. Ded. Excl., Dpto. de Informática, Facultad de Cs. Exactas, UNLP. E-mail degiusti@info.unlp.edu.ar**
**[2] Prof. Adjunto con Ded. Excl. LIDI. Dpto. de Informática, Facultad de Ciencias Exactas, UNLP. E-mail mnaiouf@info.unlp.edu.ar**
**[3] Calle 50 y 115 Primer Piso, (1900) La Plata, Argentina, Teléfono 54-21-227707. E-mail lidi@info.unlp.edu.ar**

## Introducción

En el desarrollo de la Informática actual resulta clara la importancia del procesamiento paralelo.

Tanto el procesamiento distribuido sobre arquitecturas heterogéneas soportadas por una red de comunicaciones [IEEE] [Niel90] como el multiprocesamiento sobre arquitecturas específicamente orientadas al paralelismo [Hwan93] [Leig92] son realidades que nos ofrece y nos impone la tecnología informática actual.

En este contexto resulta necesario incorporar la temática a la currícula normal de las carreras de Informática en la Universidad. De hecho, la realidad internacional indica que un porcentaje importante de los trabajos de Tesis Doctoral en Universidades "del primer mundo" giran alrededor del paralelismo.

En general en nuestras carreras hay un disímil enfoque de la enseñanza de Programación Concurrente como base necesaria para analizar la distribución del procesamiento. Por otra parte, no es homogéneo (ni en contenidos ni en intensidad) el tratamiento del tema de las arquitecturas de procesamiento, con lo que es difícil establecer el punto dentro de una currícula "standard" en que estos temas deben darse y los conocimientos previos "razonables" que tiene el alumno a esa altura.

Es muy común que hasta los '90 el tema de concurrencia fuera tratado como un "apéndice inicial" en las asignaturas de Sistemas Operativos. No era habitual tratar específicamente la programación concurrente, ni relacionarla con la arquitectura de hardware. En muchas ocasiones el paralelismo se asociaba necesariamente con una arquitectura "no-Von Neumann", alejándolo de las realidades tangibles para el alumno.

Por otra parte, la evolución tecnológica marcada por las comunicaciones y la capacidad de multiprocesamiento de los procesadores "standard" lleva a dos ejes principales del desarrollo de la Informática actual:

- El procesamiento distribuido local o remoto, con posibilidad de integrar información y recursos a través de una red con decenas a miles de procesadores [CSA90] [Coff92].

- La capacidad potencial, prácticamente de cualquier sistema de procesamiento de datos, de realizar tareas paralelas y la necesidad de desarrollar tecnología de software para explotar esta capacidad [Heer91] [Mors94].

Además, la posibilidad tecnológica de desarrollar procesadores orientados a la aplicación (como por ejemplo los DSP) permite construir multiprocesadores dedicados de altísima eficiencia, a los que hay que poder utilizar y programar adecuadamente [Moto88] [Moto90] [Moto90b].

De todo lo anterior surge la importancia de relacionar los temas de programación concurrente y paralela con la evolución tecnológica del hardware, y desarrollar conocimiento del software en las áreas de especificación, programación, verificación y evaluación de algoritmos paralelos, a fin de explotar eficientemente el recurso de hardware [Andr91] [Bust88].

Por otra parte, la potencialidad de las aplicaciones del cómputo paralelo (especialmente en el tratamiento de señales en tiempo real y en comunicaciones) hacen muy necesaria la formación de recursos humanos capacitados en el tema [Levi90] [Huss91].

En este contexto se discutió a partir de 1994 la incorporación de un Taller de Programación Paralela dentro de la Licenciatura en Informática en la Universidad Nacional de La Plata [Degi93].

En este tema, el contexto que ofrece la Licenciatura en Informática en la UNLP es relativamente diferente al de otras Universidades argentinas con carreras en Informática por 3 motivos:

**1-** Desde 1990 existe una asignatura específica de Programación Concurrente (de carácter obligatorio), y los alumnos en ese momento ya han tenido 2 cursos semestrales de Sistemas Operativos y al menos 3 cursos semestrales de Arquitecturas de procesamiento.

**2-** Hay un régimen muy flexible de optativas que permite adecuar la currícula anualmente, sin entrar en los difíciles caminos administrativos de los cambios de planes de estudio, ni en soluciones artificiales como "mutar" los contenidos de asignaturas abstractas (Computación 1, Computación 2.....Computación N).
Esto nos permitió "crear" la asignatura optativa Programación Paralela (así como Sistemas Operativos Distribuidos, Sistemas Cliente-Servidor, Software de Sistemas de Tiempo Real) sin mayores demoras ni dificultades [Info95].

**3-** Existe una línea de investigación de cierta importancia en Programación Concurrente y Paralela, con acuerdos con el exterior y Doctorandos en el país y en el exterior, que aseguran una base de trabajo de investigación y desarrollo en el tema [LIDI95].

Con estos datos previos, trataremos de analizar nuestra experiencia al poner en marcha el curso de Programación Paralela y las dificultades encontradas al momento.

## Análisis de los enfoques posibles de un curso de Programación Paralela

El primer problema que encontramos al tratar de definir el enfoque y contenido del curso fue si lo centrábamos en la Arquitectura, los Sistemas Operativos o los Lenguajes. Naturalmente esta decisión se compone de varios factores: el perfil del alumno que se está formando, la disponibilidad de docentes en el Departamento, las tendencias en el ámbito profesional en el país y en el exterior, y los recursos físicos disponibles para implementar trabajos prácticos razonables.

Claramente nos definimos por orientar el curso hacia los problemas algorítmicos y de programación de aplicaciones paralelas, por varias razones:

. El tema de arquitecturas de procesamiento paralelo (desde el punto de vista del diseño de las mismas) estaba suficientemente cubierto en la currícula por otra optativa dependiente de un grupo de investigación específico de la Facultad de Ingeniería. Allí los alumnos podían estudiar modelos de arquitecturas convencionales y no convencionales que explotan el paralelismo de hardware (normalmente con programación de bajo o muy bajo nivel).

. Orientar el curso hacia el soporte de administración de recursos distribuidos por un sistema operativo alejaba al alumno de desarrollar aplicaciones concretas competitivas [Umar93]. Normalmente pensar en desarrollar (total o parcialmente) un sistema operativo distribuido no es razonable para alumnos de un curso cuatrimestral optativo.

Por otra parte, en la tarea profesional normalmente el sistema operativo es una "precondición" existente.

. El enfoque hacia los lenguajes era muy atractivo para los alumnos, aunque tenía varios desafíos complicados: abarcar los aspectos de especificación de algoritmos concurrentes y paralelos, requerir de un soporte real para efectuar prácticas, y la necesidad de incursionar en el tema de eficiencia algorítmica y métricas del paralelismo.

Naturalmente un alumno de Informática se siente interesado por la eficiencia algorítmica y algunas aplicaciones (especialmente las de tratamiento de señales como voz e imagen en paralelo) parecían ideales para realizar trabajos prácticos.

De este conjunto de razonamientos surgió un primer plan de temas y el equipamiento básico con el que se enfrentaron las prácticas del primer curso (1994). La evolución de los temas y algunos resultados son motivo de los puntos siguientes de este trabajo.

## Tratamiento del tema de Especificación y Verificación de algoritmos

Uno de los temas de importancia fue el relacionado con la manera de especificar y verificar los algoritmos. Este era un punto en el cual debía ponerse especial atención, principalmente por la dificultad que generalmente representa el mismo para los alumnos y por la necesidad de enfatizar sobre uno de los aspectos relevantes en la formación de los mismos [Geha86].

Trabajamos muy diferentes alternativas: desde las especificaciones formales tipo CSP, CCS y Estelle, hasta ambientes orientados a la especificación y el debugging de sistemas distribuidos como STER o basados en máquinas de estado como PetriNet.

En definitiva elegimos la línea del CSP de Hoare [Hoar85] por 3 razones:

• Posibilidad de la verificación, ya que las leyes y teoremas que rigen el lenguaje permitían testear los procesos representados y las comunicaciones entre los mismos.

• Facilidad para exponer ejemplos de procesamiento distribuido en tiempo real.

• Utilización natural de CSP como paso previo hacia un lenguaje real como OCCAM, tal como describiremos más adelante.

También se analizó y utilizó una herramienta gráfica como las Redes de Petri [Pete81], las cuales poseen una concurrencia y paralelismo implícito que las convierte en útiles para la especificación, además de permitir simular el comportamiento mediante una ejecución de la red, aunque presentan algunas dificultades al momento de representar estructuras de datos.

Por esta razón, se utilizaron también extensiones de las Redes de Petri, como por ejemplo las que permiten especificar tiempos [Roze88] [Abas94] [Abas95].

Disponer de herramientas formales de especificación y verificación resulto de gran importancia en la fase de análisis y diseño de algoritmos. CSP se muestra como un lenguaje "duro" para el alumno, pero muy sólido en cuanto a la posibilidad de verificación formal y muy directo en cuanto al pasaje a OCCAM.

## Evolución del concepto de Eficiencia. Métricas del Paralelismo.

Creemos que el tema central desde el punto de vista informático en un curso de Programación Paralela es el de eficiencia algorítmica.

Por esto se estudia en el curso el concepto de speed-up; los límites teóricos alcanzables en cuanto a speed-up con arquitecturas fijas y adaptivas; el overhead de comunicaciones en el caso de procesar sobre un sistema distribuido, y la relación costo-perfomance (es decir el incremento de costo necesario para alcanzar un speed-up determinado) [Laws92] [Hwan93].

Como consecuencia natural del punto anterior se hace comprender al alumno la importancia de transformar el algoritmo, buscando maximizar el grado de paralelismo promedio, y se relaciona el modelo de grafo asociado con un algoritmo y el máximo speed-up teórico alcanzable con el mismo.

Por último se trabaja en la transformación de algoritmos secuenciales a paralelos, mostrando las técnicas y paradigmas involucrados en la transformación.

Con estos fundamentos, se insiste en el alumno en la evolución del concepto de métrica de eficiencia algorítmica y se define un parámetro de calidad del algoritmo desarrollado que comprende el speed-up alcanzado, el overhead de comunicaciones y el costo relativo a una solución convencional.

Para poder ejemplificar sobre este tema se resuelven problemas simples de filtrado de imágenes sobre arquitecturas monoprocesador, multiprocesador distribuido con PVM, transputers y se simula el comportamiento de multi-DSP [Tine96][Russ96].

## Limitaciones de las herramientas para la práctica

Una de las mayores preocupaciones fue el soporte de herramientas para la práctica. Además, existe naturalmente un "gap" entre las herramientas teóricas de especificación y modelación de los problemas y las herramientas de implementación y verificación. En síntesis, la evolución investigó diferentes líneas:

1- *Ambientes de especificación y verificación.*

Se trabajó con diversas herramientas, especialmente basadas en CSP y en Petri. Se desarrolló un ambiente con Redes de Petri extendidas que deriva un prototipo de código ejecutable en ADA, especialmente adecuado para el análisis y diseño de aplicaciones paralelas.

2- *Herramientas de soporte para procesamiento distribuido.*

Se trabajó con un soporte de comunicaciones distribuidas, basado en NetBios. Posteriormente se migró a PVM y actualmente se está estudiando MPI [PVM96] [Snir96].

En todos los casos este soporte lógico permite utilizar una red heterogénea como arquitectura MIMD para la prueba de algoritmos y las medidas de eficiencia.

3- *Lenguajes de programación convencional.*

Especialmente se utilizó ADA [Cher84] y extensiones de lenguajes conocidos por los alumnos como Pascal, C y Fortran para mostrar los mecanismos de expresión del paralelismo en distintos lenguajes.

4- *Arquitecturas centradas en transputers con lenguaje OCCAM.*

Se investigó la utilización de transputers y su programación en OCCAM [CSA90b] (también en C paralelo). Los resultados fueron varios kits de 2, 3 y 4 transputers trabajando en paralelo sobre una PC server. Los resultados desde el punto de vista didáctico han sido excelentes.

5- *Arquitecturas centradas en DSP con lenguaje C paralelo.*

En los últimos 2 años se ha avanzado (especialmente por las aplicaciones orientadas a tratamiento de voz e imagen) en la utilización de arquitecturas orientadas, basadas en DSP. Si bien ésta es una tendencia tecnológica prioritaria, no resulta tan buena didácticamente por el bajo nivel de programación que pudimos emplear (compilación en C paralelo pero código ejecutable definitivo en Assembler del DSP correspondiente) [TMS94]

6- *Trabajos en lápiz y papel sobre modelos y arquitecturas teóricas. Simuladores.*

Ha sido inevitable utilizar cálculos teóricos en la evaluación de performance de algoritmos sobre determinadas arquitecturas. También hemos utilizado simulación (especialmente para estudiar el comportamiento de DSP y multi-DSP). Sin embargo está claro que la respuesta de los alumnos no es tan buena en el caso de trabajar con un simulador.

Asimismo hemos podido utilizar maquinas especiales (en particular 2 CRAY) facilitadas por Universidades del exterior, pero es limitado el tipo de trabajos prácticos que se pueden hacer con alumnos, vía InterNet a la fecha, aunque sí se pueden hacer experiencias con los docentes.

## Algunas soluciones en la experiencia de la UNLP

Luego de 3 años el curso en La Plata está medianamente consolidado con las siguientes opciones:

*Enfoque temático.*

La idea ha sido marcar los diferentes ejes que tiene el paralelismo y priorizar el tema de la algorítmica del paralelismo, desde la especificación hasta la implementación y prueba.

Se insistió con los conceptos de eficiencia y el desarrollo de diferentes métricas del paralelismo.

*Soporte de hardware y lenguaje para la práctica.*

Se le ha dado prioridad al eje CSP-Transputers-OCCAM, sin dejar de abrir otras alternativas en los trabajos finales del alumno y en las prácticas de comparación de métricas.

En cierto modo OCCAM presenta un modelo "puro" de especificación del paralelismo que separa perfectamente la noción de proceso y procesador, permitiendo al alumno una fuerte interacción con la implementación del algoritmo.

*Soporte de evaluación de algoritmos.*

Le hemos dado gran importancia al uso de PVM por la naturalidad de su empleo por los alumnos y la posibilidad de separar los tiempos de procesamiento de los de comunicaciones, de modo de estudiar el speed-up resultante de incorporar procesadores y al mismo tiempo el overhead de utilizar un sistema físicamente distribuido.

*Tareas experimentales por los alumnos.*

Los alumnos tienen tareas individuales de investigación y desarrollo en temas teóricos y en aplicaciones.

En algunos casos exponen y esto forma parte de su evaluación. Todos deben desarrollar código paralelo y optimizarlo.

## Resultados Obtenidos y Líneas de Trabajo actuales

Los resultados de un curso universitario son muy difíciles de evaluar, o al menos tienen diferentes pautas para su evaluación. De todos modos, trataremos de sintetizar algunos resultados:

- Al tratarse de una asignatura optativa en la cual el perfil de interés de los alumnos es alto hemos tenido muy buena respuesta en el aprendizaje y la investigación de temas por los alumnos.

- Los contenidos y la bibliografía requieren un análisis permanente por la vigencia actual del tema. De todos modos, está claro que para los alumnos de la UNLP la base de arquitectura la tienen muy bien cubierta, y que en la elección entre poner el énfasis en los mecanismos de expresión y optimización del paralelismo o en la administración de recursos (centrada en el sistema operativo) hemos optado por lo primero, que se corresponde claramente con sus preferencias.

- Hemos puesto en marcha una serie de herramientas accesibles por los alumnos (Kits de transputers, simuladores DSP, NetBios, PVM, Redes de Petri extendidas, Compiladores OCCAM, C paralelo y ADA) que nos han permitido realizar prácticas y mediciones concretas, fomentando la experimentación por los alumnos.

- El sustento de un proyecto de investigación existente en el Departamento de Informática (referida a Programación Concurrente y Paralela) ha permitido orientar varios Trabajos de Grado de alumnos en esta línea, además de 2 Tesis Doctorales en temas afines.

La línea de trabajo actual supone la incorporación a partir de 1997 de un Laboratorio de Cómputo paralelo con un hipercubo de 64 transputers y 2 arreglos Multi-DSP, ambos con software de desarrollo y programación. Sobre ellos se podrá trabajar especialmente en la implementación y evaluación de algoritmos de tratamiento paralelo de señales (voz e imagen). Por otra parte se está estudiando utilizar MPI y/o PVM sobre InterNet para evaluar algoritmos masivamente paralelos.

## Conclusiones

Se ha presentado una primera implementación de un curso de Programación Paralela como asignatura optativa de Quinto Año de la Licenciatura en Informática de la UNLP, poniendo énfasis en la secuencia de decisiones respecto de contenidos, enfoque temático y recursos a utilizar que los autores fueron tomando desde 1994.

Se han planteado las dificultades más importantes al estructurar este curso y la evolución en los recursos de hardware y software empleados, marcando el tipo de equipamiento a incorporar en 1997 y la línea de investigación que sustenta la formación de docentes en el tema.

La experiencia ha sido muy positiva y coherente con las tendencias crecientes al procesamiento paralelo y más aún al procesamiento paralelo distribuido sobre arquitecturas heterogéneas que se observa en la Informática actual.

## Bibliografía


**[Abas94]** M. J. Abásolo, M. Cantarela, M. Naiouf, A. De Giusti, "Ambiente para la especificación de sistemas de Tiempo Real con Redes de Petri extendidas". Informe Técnico LIDI. Publicado en varios Congresos de la especialidad.

**[Abas95]** M. J. Abásolo, M. Cantarella, A. De Giusti, "Redes de Petri extendidas. Derivación a Código ADA". Trabajo de Tesina de Ingeniero en Sistemas UNCPBA. Expuesta en 1995.

**[Andr91]** G. Andrews, "Concurrent Programming", Benjamin/Cummings, 1991.

**[Bust88]** Bustard, Elder, Welsh, "Concurrent Program Structures", Prentice Hall, 1988.

**[Cher84]** Cherry, "Parallel programming in ANSI standard ADA", Prentice Hall, 1984.

**[Coff92]** M. Coffin, "Parallel programming- A new approach", Prentice Hall, Englewood Cliffs, 1992.

**[CSA90]** "Transputer Architecture", Computer System Architects, 1990.

**[CSA90b]** "OCCAM", Computer System Architects, 1990.

**[Degi93]** A. De Giusti, M. Naiouf, "Programación Paralela". Programa de la asignatura. Departamento de Informàtica, 1993.

**[Geha86]** N. Gehani, A. D. McGettrick, "Software Specification Techniques", Addison Wesley, 1986.

**[Heer91]** D. W. Heermann, A. N. Burkitt, "Parallel Algorithms in Computational Science", Springer-Verlag, 1991.

**[Hoar85]** C. A. R. Hoare, "Communicating Sequential Processes", Pentice-Hall, 1985.

**[Huss91]** Zahid Hussain, "Digital Image Processing", Ellis Horwood Limited, 1991.

**[Hwan93]** K. Hwang, "Advanced Computer Architecture. Parallelism, Scalability, Programmability", McGraw Hill, 1993.

**[IEEE]** Colección de "IEEE Transactions on Parallel and Distributed Systems", IEEE.

**[Info95]** Departamento deInformàtica. Plan de Estudios vigente 1995-96

**[Laws92]** H. Lawson, "Parallel processing in industrial real time applications", Prentice Hall 1992.

**[Leig92]** F. T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes", Morgan Kaufmann Publishers, 1992.

**[Levi90]** Shem-Tov Levi, A. Agrawala, "Real Time System Design", McGraw-Hill Inc, 1990.

**[LIDI95]** LIDI. Departamento de Informática. Memoria Técnica 94-95.

**[Mors94]** H. S. Morse, "Practical Parallel Computing", AP Professional, 1994.

**[Moto88]** Motorola, "DSP96002 User's Manual", Motorola, 1988.

**[Moto90]** Motorola, "Intertools. 56001", Intermetrics, 1990.

**[Moto90b]** Motorola, "Motorola DSP 56000/1", Motorola, 1990.

**[Niel90]** K. Nielsen, "Ada in Distributed Real-Time Systems", McGraw-Hill, 1990.

**[Pete81]** J. Peterson, "Petri Nets Theory and the Modeling of Systems", Prentice-Hall, 1981.

**[PVM96]** "Parallel Virtual Machine", World Wide Web.

**[Roze88]** Rozenberg (Ed.), "Advances in Petri Nets", Springer-Verlag, 1988.

**[Russ96]** C. Russo, H. Ramón, A. Anderson, D. Dirazar, A. De Giusti, "Algoritmo Distribuido de Compresión de Datos. Una Experiencia Comparativa con Sockets y PVM", enviado al 2° CACIC, 1996.

**[Snir96]** M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, "MPI: The Complete Reference", The MIT Press, 1996.

**[Tine96]** F. Tinetti, C. Russo, H. Ramón, A. De Giusti, "Analysis and Extension of a Simulation Tool for Multi-DSP Parallel Processing", enviado al 2° CACIC, 1996.

**[TMS94]** "TMS320C3x User's Guide.", Texas Instruments, 1994.

**[Umar93]** A. Umar, "Distributed Computing and Client-Server Systems", P T R Prentice Hall, 1993