UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

PROGRAMACIÓN CONCURRENTE

Año 2010

Carrera: ***Licenciatura en Informática***
Plan 2003-2007
Licenciatura en Sistemas
Plan 2003-2007

Año: ***3°***
Duración: ***Semestral***
Profesor: ***Dr. Marcelo Naiouf***
Lic. Franco Chichizola
Hs. semanales: **6 hs.**

OBJETIVOS GENERALES:

Plantear los fundamentos de la programación concurrente, estudiando su sintaxis y semántica, así como herramientas y lenguajes para la resolución de programas concurrentes.

Analizar el concepto de *sistemas concurrentes* que integran la arquitectura de hardware, el sistema operativo y los algoritmos que permiten la resolución de problemas concurrentes.

Estudiar los conceptos fundamentales de comunicación y sincronización entre procesos, por memoria compartida y por mensajes.

Vincular la concurrencia en software con los conceptos de procesamiento distribuido y paralelo, para tener los conceptos de soluciones multiprocesador con algoritmos concurrentes.

CONTENIDOS MINIMOS:

- Especificación de la ejecución concurrente
- Comunicación y sincronización.
- Concurrencia con variables compartidas.
- Concurrencia con pasajes de mensajes.
- Lenguajes de programación concurrente.
- Introducción a los conceptos de procesamiento paralelo.

Programa

1. Conceptos básicos

Objetivos de los sistemas concurrentes.
Procesamiento secuencial, concurrente y paralelo. Características.

Evolución histórica.
Procesos. Programa concurrente. No determinismo.
Clases de aplicaciones. Multithreading, Cómputo paralelo y distribuido.
Concurrencia y paralelismo.
Algoritmos concurrentes, distribuidos y paralelos.
Areas de estudio en sistemas concurrentes.
Relación con la arquitectura. Monoprocesadores. Multiprocesadores: Clasificaciones y ejemplos. Conceptos de arquitecturas Grid. Memoria compartida distribuida.
Relación con el sistema operativo. Requerimientos para el sistema operativo.
Relación con el lenguaje. Requerimientos para el sistema operativo.
Sincronización y comunicación.
Sincronización por exclusión mutua y por condición.
Comunicación por memoria compartida y por mensajes.
Prioridad, granularidad, deadlock, manejo de recursos.
Paradigmas de resolución de programas concurrentes: iterativo (ej, multiplicación de matrices), recursivo o *divide & conquer* (ej. el problema de la cuadratura), *pipeline* o productor consumidor, cliente/servidor y sus variantes, *peers* o pares interactuantes

2. *Concurrencia y sincronización*

Aspectos de programación secuencial
Especificación y semántica de la ejecución concurrente. La sentencia *co* y *process*
Acciones atómicas y sincronización.
El problema de interferencia. Historias válidas e inválidas.
Atomicidad de grano fino y de grano grueso.
La propiedad de “a lo sumo una vez”.
La sentencia Await. Semántica. Especificación de la sincronización.
Técnicas para evitar interferencia.
Propiedades de seguridad y vida.
Políticas de scheduling y Fairness.
Requerimientos para los lenguajes de programación.
Problemas en sistemas distribuidos.

3. *Concurrencia con variables compartidas*

Sincronización por variables compartidas
Sincronización de grano fino.
Secciones críticas (SC). Definición del problema. Propiedades necesarias de las soluciones.
Soluciones de tipo spin-locks al problema de la SC.
Algoritmos clásicos de soluciones fair al problema de la SC (tie-breaker, ticket, bakery).
Implementación de sentencias Await arbitrarias.
Sincronización barrier. Definición.

Soluciones posibles (contador compartido, flags y coordinadores, árboles, barreras simétricas, butterfly).
Algoritmos *data parallel*.. Computación de prefijos.

Sincronización por semáforos
Defectos de la sincronización por variables compartidas.
Semáforos. Sintaxis y semántica.
Usos básicos y técnicas de programación.
Soluciones a SC y barreras.
Semáforos binarios divididos (*split*).
Exclusión mutua selectiva.
La técnica "*passing the baton"*. Definición y aplicaciones.
Sincronización por condición general.
Alocación de recursos. SJN.
Ejemplos clásicos: filósofos, lectores y escritores, productores y consumidores con buffer limitado, etc.
Semáforos en lenguajes reales: Pthreads. Ejemplos.

Sincronización por monitores
Noción de Regiones Críticas Condicionales.
Monitores. Sintaxis y semántica.
Sincronización en monitores.
"Signal and wait" y "Signal and continue": diferencias.
La técnica "*passing the condition".*
Ejemplos clásicos: buffer limitado, lectores y escritores, alocación, etc.
Diseño de un reloj lógico. Alternativas.
El problema del peluquero: rendezvous.
Scheduling de discos. Ejemplo. Enfoques alternativos para la sincronización.
Monitores en lenguajes reales: Java, Pthreads. Ejemplos.

Implementaciones
Conceptos de implementación de procesos en arquitecturas mono y multiprocesador.
Kernel monoprocesador y multiprocesador.

4. *Programación distribuida. Concurrencia con pasaje de mensajes*

Programas distribuidos.
Relación entre mecanismos de comunicación.
Clases básicas de procesos: productores y consumidores, clientes y servidores, peers.
Control de concurrencia en Sistemas Distribuidos.

Mensajes asincrónicos
Sintaxis y semántica. Canales. Operaciones.
Filtros. Redes de Filtros.

Clientes/Servidores. Algoritmos clásicos. Monitores activos. Continuidad conversacional.
Peers. Intercambio de valores. Cálculo de la topología de una red.
Mensajes asincrónicos en lenguajes reales MPI. Extensión de lenguajes secuenciales con bibliotecas específicas.

Mensajes sincrónicos
Sintaxis y semántica.
Conceptos de CSP.
Comunicación guardada. Sintaxis y semántica.
Filtros. Clientes y servidores. Asignación de recursos.
Interacción entre procesos paralelos.
Ejemplos.
Mensajes sincrónicos en lenguajes reales: OCCAM. Constructores secuenciales y paralelos. Comunicación y sincronización. Ejemplos

Linda: extensiones para el manejo de PM
Linda. Estructuras de datos distribuidas
El concepto de *bag of tasks*.
Espacio de tuplas e interacción entre procesos.
Ejemplos

Remote Procedure Calls y Rendezvous.
Sintaxis y semántica.
Similitudes y diferencias.
RPC: Sincronización en módulos.
Discusión revisada de aplicaciones.
RPC en lenguajes reales: Java. RMI. Ejemplos.
Rendezvous en lenguajes reales: Ada. Tasks y sincronización. Ejemplos

Primitivas múltiples
La notación de primitivas múltiples.
Sintaxis y semántica.
MPD. Componentes de programa. Comunicación y sincronización. Ejemplos

Implementaciones
Conceptos de implementación de mecanismos de comunicación y sincronización en ambientes distribuidos.

5. *Overview de lenguajes de programación*

Resolución de problemas mediante diferentes paradigmas de interacción entre procesos: Servidores replicados, algoritmos heartbeat, algoritmos pipeline, prueba-eco, broadcast, token passing, manager/workers.
Ejemplos. Comparación de alternativas.

6. ***Introducción a la Programación Paralela***
Computación científica.
Necesidad del paralelismo.
Diseño de algoritmos paralelos.
Métricas. Conceptos de speedup y eficiencia.
La ley de Amdahl y la ley de Gustafson.
Concepto de escalabilidad.

En la práctica se realiza trabajo experimental sobre arquitecturas multiprocesador distribuidas, tales como clusters y multiclusters de PCs, y con multiprocesadores con memoria compartida.

Metodología de enseñanza

La actividad curricular se organiza en: clases teóricas, las cuales imparten los temas de la asignatura; explicaciones de práctica, que actúan a modo de articulación entre la teoría y la práctica y donde se plantean y resuelven problemas “tipo”; clases prácticas donde los alumnos trabajan sobre los ejercicios propuestos en la guía de trabajos prácticos y clases de consulta (de teoría y práctica).
Periódicamente se publican cuestionarios de teoría a modo de guía a fin de que los alumnos reflexionen sobre los puntos más importantes.

Propuesta de evaluación

Para obtener la cursada el alumno debe aprobar un examen parcial para el que dispone de una fecha y dos recuperatorios. Para la aprobación final de la asignatura debe aprobar un examen final teórico-práctico en mesa de finales.
Existen modalidades opcionales para la obtención de cursada y final.
a) para la cursada: aprobación de un parcial reducido cuya precondición es aprobar al menos 3 de 4 parcialitos prácticos durante la cursada (si se desaprueban se puede rendir el parcial normal)
b) para el final, cumpliendo las siguientes condiciones (c/u tiene como precondición la anterior):
1) Rendir al menos 3 de 4 parcialitos teóricos (No es obligatorio aprobarlos)
2) Realizar y aprobar un trabajo sencillo en máquina, asignado por grupos de 2 alumnos al finalizar las prácticas
3) Aprobar un parcial teórico
4) Realizar un trabajo individual de mayor complejidad que 2), que debe ser defendido en un coloquio en fecha de final

REFERENCIAS BIBLIOGRAFICAS

- Andrews G.. “Foundations of Multithreaded, Parallel and Distributed Programming”, Addison Wesley, 2000
- Ben-Ari, M. “Principles of Concurrent and Distributed Programming, 2/E”. Addison-Wesley. 2006. ISBN 0-321-31283-X
- Grama A., Gupta A., Karypis G., Kumar V., "An Introduction to Parallel Computing. Design and Analysis of Algorithms", Pearson Addison Wesley, 2nd Edition, 2003
- Ghosh, Sukimar. “Distributed Systems: An Algorithmic Approach”. Chapman & Hall/CRC, 2007. ISBN1584885645, 9781584885641

- Akl S., “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.
- Brinch Hansen, P., “Studies in Computational Science. Parallel Programming Paradigms”, Prentice Hall, 1995.
- Chandy, Misra, “Parallel Program Design. A Foundation”, Addison Wesley, 1988.
- Chiola G., G. Ciaccio, “Lightweigth Messaging Systems”, in R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 246-269, 1999.
- Downey, Allen. “The Little Book of Semaphores, Second Edition”. Free book disponible en http://www.freetechbooks.com/the-little-book-of-semaphores-second-edition-t519.html, 2007.
- Habermann A., Perry D., “Ada for Experienced Programmers”, Addison-Wesley Publishing Company, Inc. 1993
- Hoare C., “Communicating Sequential Processes”, Englewood Cliffs, Prentice Hall, 1985
- Hwang K., Xu Z., “Scalable Parallel Computing”, McGraw-Hill, 1998.
- Jordan H.F., Alaghband G., Jordan H.E., "Fundamentals of Parallel Computing", Prentice Hall, 2002
- Leopold C., "Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches", Wiley Series on Parallel and Distributed Computing. Albert Zomaya Series Editor, 2001
- Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. *MPI: The Complete Reference*. Cambridge, MA: MIT Press, 1996. Available in web site: http://www.netlib.org/utk/papers/mpi-book/mpi-book.html
- Tinetti F., De Giusti A., "Procesamiento Paralelo. Conceptos de Arquitectura y Algoritmos", Editorial Exacta, 1998.
- Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networking Workstations, Prentice-Hall, Inc., 1999.