PROGRAMACIÓN CONCURRENTE

Año 2014

Carrera/ Plan:
Licenciatura en Informática
Plan 2003-07 / Plan 2012
Licenciatura en Sistemas
Plan 2003-07 / Plan 2012
Analista Programador Universitario
Plan 2007

Año: *3°*

Régimen de Cursada: *Semestral*
Carácter: Obligatoria

Correlativas: Introducción a los Sistemas Operativos- Seminario de Lenguajes- Taller de Lecto-comprensión y Traducción en Inglés

Profesor: *Marcelo Naiouf, Franco Chichizola, Laura De Giusti*

Hs. semanales: 6 hs.

FUNDAMENTACIÓN

La temática de la Concurrencia es central en el desarrollo actual de la Ciencia Informática, en particular por el creciente desarrollo de arquitecturas multiprocesador que permiten implementar físicamente los conceptos teóricos de concurrencia "real".
El impacto de la concurrencia se refleja en diferentes ámbitos de la disciplina tales como las arquitecturas, los sistemas operativos, los lenguajes y el diseño y desarrollo de aplicaciones. En este sentido, se impone que los futuros profesionales sean capaces de desarrollar soluciones que utilicen adecuadamente la tecnología disponible con fundamentos teóricos firmes.

OBJETIVOS GENERALES:

Brindar los conceptos fundamentales de Concurrencia en software. Analizar la semántica y sintaxis para especificar concurrencia. Analizar el concepto de sistema concurrente compuesto por la arquitectura, el sistema operativo y los algoritmos. Estudiar la sincronización de procesos concurrentes por memoria compartida y mensajes. Vincular la concurrencia en software con los conceptos de procesamiento distribuido y paralelo. Desarrollar estudios de casos con diferentes lenguajes/ herramientas para concurrencia.

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 ANALÍTICO

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 y Cloud. Memoria compartida distribuida.
Relación con el sistema operativo. Requerimientos para el sistema operativo.
Relación con el lenguaje. Requerimientos para el lenguaje.
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, recursivo o *divide & conquer*, *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.
Planteo de soluciones (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. Políticas de alocación. SJN.
Ejemplos clásicos: filósofos, lectores y escritores, productores y consumidores con buffer limitado, etc.
Semáforos en lenguajes reales: Pthreads. Ejemplos.

Sincronización por monitores
Evolución histórica a partir de semáforos. Conceptos de regiones críticas condicionales.
Monitores. Sintaxis y semántica.
Sincronización en monitores.
Disciplinas de señalización: "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. Concepto de 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
El concepto de *bag of tasks*.
Linda. Estructuras de datos distribuidas
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. *Paradigmas de interacción entre procesos distribuidos*
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*
Objetivos del cómputo paralelo.
Computación científica.
Necesidad del paralelismo. Areas de aplicación.
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 (clusters), multiprocesadores con memoria compartida e híbridos.

METODOLOGÍA DE ENSEÑANZA

La actividad curricular se organiza en:
- clases teóricas, a cargo de un profesor (en horario de mañana y de tarde), en las cuales se imparten los temas de la asignatura
- explicaciones de práctica, a cargo de un profesor, y que actúan a modo de articulación entre teoría y práctica y donde se plantean y resuelven problemas "tipo".
- clases prácticas, a cargo de auxiliares docentes (ayudantes coordinados por JTP, en horario de mañana, tarde y sábados a la mañana), donde los alumnos trabajan sobre los ejercicios propuestos en la guía de trabajos prácticos
- 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.

El reglamento y cronograma tentativo son conocidos por los alumnos desde el inicio de la actividad curricular.

Se utiliza un entorno virtual de enseñanza-aprendizaje (WebUNLP), donde se encuentran disponibles clases, guías de TP, avisos, resultados de exámenes, etc.

Para las clases teóricas y las explicaciones de práctica se utiliza PC, cañón y pizarrón.

Los alumnos pueden realizar prácticas en PC usando distintos lenguajes/bibliotecas que soportan concurrencia.

7

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 de promoción 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 examen parcial)
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) Aprobar un parcial teórico
3) Realizar un trabajo individual, que se asigna luego de aprobar el parcial teórico y debe ser defendido en un coloquio en fecha de final (antes de abril de 2017).

BIBLIOGRAFÍA OBLIGATORIA

- 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
- Filminas de las clases teóricas.
- Naiouf, De Giusti A., De Giusti L, Chichizola, "Conceptos de concurrencia y paralelismo", UNLP, a publicar 2014.

BIBLIOGRAFÍA COMPLEMENTARIA

- Barnes J., "Programming in Ada 2005 with CD", Addison Wesley, 2006.
- Brinch Hansen, P., "Studies in Computational Science. Parallel Programming Paradigms", Prentice Hall, 1995.
- Chandy, Misra, "Parallel Program Design. A Foundation", Addison Wesley, 1988.
- 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.
- Goetz B., Peierls T., Bloch J., Bowbeer J., "Java Concurrency in Practice", Addison Wesley, 2006.
- Herlihy M., Shavit N., "The Art of Multiprocessor Programming". Morgan Kaufmann, 2008. (Revised reprint, 2012)
- Hoare C., "Communicating Sequential Processes", Englewood Cliffs, Prentice Hall, 1985

- Jordan H.F., Alaghband G., Jordan H.E., "Fundamentals of Parallel Computing", Prentice Hall, 2002
- Pacheco, P. "An introduction to parallel programming". Morgan Kaufmann, 2011.
- Raynal M. "Concurrent Programming: Algorithms, Principles, and Foundations". Springer, 2012.
- Taubenfeld, Gadi. "Synchronization Algorithms and Concurrent Programming". Prentice Hall. 2006.

CRONOGRAMA DE CLASES Y EVALUACIONES

El comienzo de clases está previsto para la semana del 11 de agosto.
El cronograma detallado se pone en conocimiento de los alumnos al inicio del curso.
El esquema preliminar por bloque de las clases y evaluaciones es el siguiente:

Clase	Contenidos/Actividades	Evaluaciones previstas
Clases 1 y 2	Conceptos básicos. Comunicación y sincronización. Interferencia.	
Clases 3 a 6	Concurrencia con memoria compartida.	Parcialito 1
Clases 7 a 9	Concurrencia con memoria distribuida	Parcialito 2
Clase 10	Introducción a la Programación Paralela.	Parcialito 3
		Parcialito 4
Semana 3/11		Parcial
Semana 24/11		1er recuperatorio
Semana 9/12		2do recuperatorio
10/3/2014		Parcial teórico

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos):

Plataforma virtual: webunlp.unlp.edu.ar
Web: weblidi.info.unlp.edu.ar/catedras/concurrente/WEB/
Mail: mnaiouf@lidi.info.unlp.edu.ar

Firmas del/los profesores responsables: