DEGREE CURRICULUM

CONCURRENT AND PARALLEL SYSTEMS

Academic year 2014-15

CONCURRENT AND PARALLEL SYSTEMS 2014-15

Subject's general information

Subject name	Concurrent and Parallel systems
Code	102022
Semester	1r Q Continuous Assessment
Typology	Mandatory
ECTS credits	6
Theoretical credits	3
Practical credits	3
Office and hour of attention	Monday 16h-17h (s3/17) Thursday 12h-13h (s3/17)
Department	Informàtica i Enginyeria Industrial
Teaching load distribution between lectures and independent student work	60 classroom hours 90 homework hours
Modality	Presencial
Important information on data processing	Consult this link for more information.
Language	Spanish
Degree	Degree in Computer Enginneering
Distribution of credits	Fernando Cores 6.0
Office and hour of attention	Monday 16h-17h (s3/17) Thursday 12h-13h (s3/17)
E-mail addresses	fcores@diei.udl.cat

CONCURRENT AND PARALLEL SYSTEMS 2014-15

Fernando Cores Prado

Subject's extra information

The computer industry is suffering, if not another revolution undoubtedly vigorous stirring. The main processor manufacturers have resigned, for the moment, to try to design / manufacture processors run faster. Moore's Law has not yet been repealed: each year, more and more transistors fit in the same space, but its clock speed cannot be increased due to overheating and the high-consumption associated with high clock frequencies. Instead of increasing the speed of processors, manufacturers have chosen architectures "multi-core", where multiple processors (cores) are integrated together within a chip, which can communicate directly through caches shared. Multiprocessor chips perform calculations more efficiently by exploiting parallelism: the use of multiple processors to work on a single task.

The spread of multiprocessor architectures will have a widespread impact on how it designs and develops software for the future. Until recently, advances in technology also meant advances in clock speed, so the software actually accelerated its execution speed at half time passed now; however, this automatic upgrade of the applications is over. Advances in technology will mean greater parallelism and not increase the clock speed, thus exploiting this parallelism is one of the biggest challenges of modern computer pending.

This course focuses on the design of concurrent and parallel applications that harness the processing power of multiprocessor systems to improve application performance. In developing these applications programming challenges arise at all levels of the multiprocessor system - on a very small scale, within the nuclei of the same chip, you need to coordinate access to the same position shared memory, and larger scale is required coordinate the routing of data between supercomputer processors. Concurrent and parallel programming represents a major challenge because modern computer systems are inherently asynchronous: activities can be stopped or delayed without notice interruptions, preference, cache misses, exceptions, and other events. These delays are inherently unpredictable and can vary greatly in length, causing the behaviour of a concurrent / parallel application cannot be deterministic (different versions with the same input parameters can generate different results) if not programmed correctly.

In this course, we will study concurrent programming techniques, especially suitable for maximum performance multiprocessor systems. Let's focus on the different models to express concurrency, the use of threads for the design of concurrent applications, synchronization mechanisms both low (lights) and high (monitors and condition variables) to coordinate concurrent tasks application and different high-level APIs that facilitate the programming of these applications.

Learning objectives

- To understand the importance of concurrent programming into existing applications.
- To identify the main characteristics of the different types of concurrent systems in the world.
- To know and understand the problems posed by the development of concurrent programs that do not appear in sequential programming.
- To understand the concepts of synchronization and mutual exclusion between processes.
- To identify the security properties and vivacity that a concurrent system must meet and be able to reason if these properties are met.
- To understand the main models of concurrent, parallel and distributed programming.
- To gain experience and knowledge in communication and synchronization mechanisms that are used today to develop concurrent programs for both shared memory systems to distributed systems.
- To understand the operation of traffic lights and monitors and synchronization mechanisms for shared memory and understand how they can solve problems of concurrent programming using monitors.
- To apply methodologies of software engineering in the development of concurrent and parallel applications.

Competences

Degree Competences:

- EPS7. Capacity to work in situations with a lack of information and/or under pressure.

Specific Competences:

- GII-CRI11. Knowledge and application of the characteristics, functionalities and structure of the Distributed Systems, the Networks of Computers and Internet and design and implement applications based in them.
- GII-CRI14. Knowledge and application of the basic principles and basic techniques of the parallel, concurrent, distributed and of real time programming.
- GII-CRI16. Knowledge and application of the principles, methodologies and life cycle of the software engineering.

Subject contents

1. **Introduction to the concurrency**
    1. Concurrency definition
    2. Need and benefits of concurrent programming
    3. Concurrent hardware architectures
    4. Features of concurrent systems
    5. Concurrent programs Specification
        1. Conditions of Bernstein
        2. Concurrency Table
        3. Precedence Graphs
    6. How to express concurrency
        1. Cobegin/Coend
        2. Fork/Join
        3. Unix + C
        4. Examples
    7. Case study: Linux and Java threads
2. Design concurrent and parallel applications
    1. Model of concurrent/parallel programming
    2. Efficiency of concurrent/parallel programs
        1. Scope of parallelism
        2. Granularity
        3. Location
    3. 2.3. Design of concurrent programs
        1. Stages of Design
        2. Task decomposition techniques
        3. Parallel design patterns
        4. Distribution and communication of tasks
    4. Case studies
3. **Synchronization of concurrent processes**
    1. Synchronization Introduction
        1. Race conditions
    2. The problem of mutual exclusión
        1. Software Solutions
        2. Hardware Solutions
    3. Conditional synchronization
        1. Classical problems of synchronization
    4. Case Study:
        1. Pthread mutex and condition variables
        2. Java Synchronization
4. **High-level APIs for concurrency and synchronization**
    1. Monitors
    2. Patterns concurrent
    3. Case Study:
        1. Boost Library
        2. Concurrent Java API

Methodology

The course is eminently practical and therefore will give more weight to the practices and programming. Basically we work with three languages, C for concurrent programming with threads and Java APIs to introduce high-level synchronization and concurrent patterns. None of these languages will be explained from scratch, because both C and Java have already seen in previous subjects of the degree.

To continue the course is essential for students to have a good foundation of programming in C and Java. It is much harder, learn to develop concurrent and parallel programs if not previously sequential programming dominates. In the course it is assumed that students are able to design, develop and debug sequential applications of medium difficulty without much trouble.

Learning outcomes.

- To be able to design and develop algorithms based on shared memory and distributed systems to solve problems that require concurrent and parallel programming systems.
- To understand and be able to use libraries and standardized implementation of concurrent programs based on shared memory platforms.

Development plan

Week	Description	Classroom Activity GG	Classroom Activity GM	HTP (4 Hrs)	Autonomous work activity	HTNP (6 Hrs)
1	Presentation Concurrency Introduction	Subject presentation T1: Concurrency Introduction	**L1: Posix Threads**	4	**Study literature and the program**	2
2	Concurrency Introduction	T1: Concurrency Introduction	**L1: Posix Threads** **L2: Java Threads** Practice 1: Presentation	4	T1: Concurrency Introduction	4
3	Concurrency Introduction	T1: Concurrency Introduction T2: Design concurrent applications	**L2: Java Threads**	4	Practice1 Problems: Concurrency	6
4	Design concurrent applications	T2: Design concurrent applications	**L2: Java Threads** Problems: Design	4	Practice 1 T2: Design	5
5	Design concurrent applications	T2: Design concurrent applications	Problems: Design	4	Practice 1 Problems: Design	6
6	Design concurrent applications	T2: Design concurrent applications	Practice 1: Delivery Correction P1 Eval. Cont.	4	Practice 1 Problems: Design	5
7	Synchronization	T3: Synchronizing concurrent processes Problems: Synchronization	**L3: Synchronization Linux** Practice 2: Presentation	4	T3: Synchronization	6
8	Synchronization	Problems: Synchronization 1st Exam Review	**L4: Synchronization Java**	4	Practice 2 T3: Synchronization	7
9		**1er Partial**			Study	6

10	Synchronization	Partial correction 1 T3: Synchronizing concurrent processes	Problems: Synchronization	4	Practice 2 T3: Synchronization	4
11	Synchronization	T3: Synchronizing concurrent processes	Problems: Synchronization	4	Practice 2 Problems: Synchronization	8
12	Synchronization	T3: Synchronizing concurrent processes	Practice 2: Delivery	4	Practice 2 Problems: Synchronization	7
13	Concurrency & synchronization High-level APIs	T4: High-level APIs for concurrency and Synchronization	Practice 3: Presentation **L4: Boost Library**	4	Problems: Synchronization T4: High-level APIs	7
14	Concurrency & synchronization High-level APIs	T4: High-level APIs for concurrency and Synchronization	Correction P2 Eval. Cont.	4	Practice 3 T4: High-level APIs	7
15	Concurrency & synchronization High-level APIs	T4: High-level APIs for concurrency and Synchronization 2nd Exam Review	Practice 3: Delivery	4	Practice 3 T4: High-level APIs	7
16		**2on Partial**		2	Study	6
17						
18						
19		Recovery		2		5

Evaluation

The course is approved on a final note exceeding 5 having completed all laboratory practice (each with at least 4 note)

The final grade for the course is derived from the weighted sum of the grades of the 2 tests and practical notes more class participation and continuous evaluation.

The course has two sets, each with a weight of 25% of the final grade. These tests are mandatory and removed matter.

The completion and improvement of laboratory practices is mandatory to pass the course. The practices will be evaluated with a note that represents 40% of the final grade for the course. The copy of any of the practices of the course will involve suspend all practice and thus the subject.

Table. Assessment Activities

Assessment activity	Weighting	Low Mark	In group	Mandatory
1st Partial Exam	25%	4	NO	YES
2nd Partial Exam	25%	4	NO	YES
Practices	40%	5	Yes (≤2)	YES
Problems	10%	NO	Yes (≤2)	NO

Class participation	1 point	NO	NO	NO

There minimum mark of 4 in the middle of the written tests to obtain the final grade for the course.

There is also a high note (5) to the labs. A practice is considered suspended if fails to arrive to mark of 4.

Bibliography

Textbooks:

[Pal03] José Tomás Palma Méndez, M. C. Garrido Carrera, F. Sanchez Figueroa, A. Quesada Arencibia, "Programación Concurrente ", Thomson, 2003.
[Her08] Maurice Herlihy, Nir Shavit, "The Art of Multiprocessor Programming", Morgan Kaufmann, 2008.
[Dou00] Douglas Lea, "Concurrent Programming in Java: Design Principles and Patterns", Addison-Wesley Professional, 2000.
[Pac11] Peter Pacheco, "An Introduction to Parallel Programming", Morgan Kaufmann, 2011.

Extended Bibliography:

[Gad06] Gadi Taubenfeld, "Synchronization Algorithms and Concurrent Programming", Pearson / Prentice Hall, 2006
[Ben06] M. Ben-Ari, "Principles of Concurrent and Distributed Programming", Addison-Wesley, 2nd Edition, 2006
[Nic96] B. Nichols, D. Buttlar, "Pthreads Programming", O'Reilly, 1996
[Kay97] KayA. Robbins, Steven Robbins, "UNIX Programación Práctica. Guía para la Concurrencia, la Comunicación y los Multihilos", Edt.Prentice-Hall, 1997.
[Afz97] Afzal, Amir, "Intoducción a UNIX un enfoque práctico ", Edt.Prentice-Hall,