Course guide
270602 - CPDS - Concurrence, Parallelism and Distributed Systems

Last modified: 13/07/2022

Unit in charge: Barcelona School of Informatics
Teaching unit: 701 - DAC - Department of Computer Architecture.
723 - CS - Department of Computer Science.

Degree: MASTER'S DEGREE IN INNOVATION AND RESEARCH IN INFORMATICS (Syllabus 2012). (Compulsory subject).

Academic year: 2022 **ECTS Credits:** 6.0 **Languages:** English

LECTURER

Coordinating lecturer: JORDI GUITART FERNANDEZ

Others: Primer quadrimestre:
JORGE CASTRO RABAL - 11, 12
JOAQUIN GABARRÓ VALLÉS - 11, 12
JORDI GUITART FERNANDEZ - 11, 12
JOSE RAMON HERRERO ZARAGOZA - 11, 12

PRIOR SKILLS

Concurrency: Java at the level of classes and objects.
Parallelism: basic understanding of parallel architectures, including shared- and distributed-memory multiprocessor systems.
Distribution: understanding of the internal structure and operation of an operating system and computer networks.

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.
CEE2.3. Capability to understand models, problems and mathematical tools to analyze, design and evaluate computer networks and distributed systems.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.

Generical:
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.

Transversal:
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.
CTR6. REASONING: Capacity for critical, logical and mathematical reasoning. Capability to solve problems in their area of study. Capacity for abstraction: the capability to create and use models that reflect real situations. Capability to design and implement simple experiments, and analyze and interpret their results. Capacity for analysis, synthesis and evaluation.

TEACHING METHODOLOGY

During the course there will be two types of activities:

a) Activities focused on the acquisition of theoretical knowledge.
b) Activities focused on the acquisition of knowledge through experimentation by implementing and evaluating empirically in the laboratory the mechanisms explained at a theoretical level.

The theoretical activities include participatory lecture classes, which explain the basic contents of the course. The practical activities include seminar laboratories where students implement the mechanisms described in the lectures. The seminars require a preparation by reading the statement and supporting documentation, and a further elaboration of the conclusions in a report.

LEARNING OBJECTIVES OF THE SUBJECT

1.Understand the definition of distributed system and its possible applications, as well as challenges to be faced to design and implement it.
2.Understand the problem of time and events ordering in a distributed system and explain and implement the mechanisms of logic clocks to attack this problem and algorithms to synchronize physical clocks in a distributed system.
3.Understand the problem of obtaining a consistent global state in a distributed system and explain the distributed snapshot mechanism to attack this problem, as well as define predicates for global assessment of properties in a distributed system.
4.Describe, compare and implement algorithms for the coordination of processes in a distributed system, including the coordination necessary for leader election, multicast group communication and consensus.
5.Understand the problem of concurrent execution of transactions and describe, compare, and implement different concurrency control mechanisms.
6.Extend the concept of transaction, the commit protocol, and the concurrency control mechanisms in a distributed system.
7.Understand the application of replication in a distributed system, as well as the consistency problems introduced, and describe the corresponding consistency models and their implementation.
8.Understand the problem of concurrent access to resources by different agents (threads, processors), and the design principles to ensure a correct coordination to avoid deadlocks
9.Understand the design of concurrent programs for shared memory architectures
10.Understand the design of concurrent programs for message-passing architectures
11.Understand and measure the potential parallelism available in a sequential application and the performance achieved with its parallel implementation
12.Decide the most appropriate decomposition strategy to express parallelism in an application
13.Specify, using the appropriate programming paradigm, an efficient parallel version that corresponds to a certain task/data decomposition

STUDY LOAD

Type	Hours	Percentage
Hours large group	26,0	17.22
Self study	96,0	63.58
Guided activities	3,0	1.99
Hours small group	26,0	17.22

Total learning time: 151 h

CONTENTS

Transition systems and process algebra

Description:
Transition systems and primitive processes. Operations prefix and choice. Concurrent processes and their implementation. Description of the LTS system. Implementation in Java and Erlang.

Understanding parallelism

Description:
Introduction to parallel architectures: shared- and distributed-memory architectures and accelerators. Task graph: nodes and edges. Metrics. Speed-up and efficiency. Amdahl law.

Concepts underlying distributed systems

Description:
Definition of a distributed system. Potential applications of a distributed system. Examples of distributed systems. Challenges to design and implement a distributed system: heterogeneity, security, lack of global view, concurrency, lack of a single point of control, asynchrony, openness transparency, fault tolerance, scalability.

Safety and liveness properties

Description:
Description and examples of safety properties and deployment in LTS. Description of liveness properties, especially the progress property and implementation in LTS.

Concurrent objects, mutual exclusion and synchronization conditions. The deadlock problem.

Description:
Problem of destructive interference. Locks and mutual exclusion. Modeling of traffic lights and monitors and of the problem of overlapping. Problem of deadlock, analysis by LTS.

Message passing. Architectures

Description:
Message passing. Client / server architecture, other architectures Introduction to.: Pilelines filters, suvervisor / workers, advertiser / listener. Message-passing in Erlang. Design of Erlang in a client / server architecture.

Predicting and analyzing performance

Description:
Use of models and tools to understand parallelism and performance (Tareador, Extrae, Paraver and Dimemas).

Distributed-memory programming using MPI

Description:
Cluster architecture overview. Process creation, identification and termination. Point-to-point vs. collective operations. Synchronous vs. asynchronous communications.

Programming GPU devices for computation acceleration using CUDA

Description:
GPU architecture overview. Decompositions suitable for GPU acceleration. CUDA programming principles. CUDA Parallel Execution Model: host and device.

Distributed algorithms: Time, global state, coordination, and agreement

Description:
Time and events ordering in a distributed system. Logical clocks: happened-before relation, Lamport logical clocks (scalar, vector). Algorithms to synchronize physical clocks in a distributed system: Cristian (NTP), Berkeley. Consistent global state in a distributed system. The Chandy and Lamport's mechanism of distributed snapshot. Global predicates for evaluating properties in a distributed system: properties of predicates (stability), occurrence of predicates (possibly, definitely). Coordination of processes in a distributed system for the election of leader: Bully, Ring. Coordination of processes in a distributed system for multicast group communication: basic reliable multicast, scalable reliable multicast, ordered multicast (FIFO, causal, total), atomic multicast. Coordination of processes in a distributed system to ensure consensus: the two army problem, Dolev & Strong's algorithm, the Byzantine generals problem, Paxos

Distributed shared data: Transactions, consistency, and replication

Description:
Concurrent execution of transactions: lost update, inconsistent retrievals, serial equivalence, recovery of aborts (dirty read, write premature). Concurrency control mechanisms: two-phase locking (including deadlock detection and treatment), optimistic concurrency control, timestamp ordering. Distributed transaction. Distributed commit protocols: one and two-phase. Concurrency control mechanisms in a distributed system: two-phase locking (including distributed deadlock detection and treatment), optimistic concurrency control, timestamp ordering. Replication and consistency in a distributed system. Data-centered strong consistency models: strict, linearizability, sequential. Data-centric relaxed consistency models: usage of synchronization variables. Client-centric consistency models: eventual, monotonic-read, monotonic-write, read-your-writes, writes follow-reads. Implementations of consistency models: primary-based protocols (remote-write, local-write) and replicated-write protocols (active replication, quorum-based protocols)

ACTIVITIES

Concurrency, parallelism and distribution: fundamental concepts

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents and their subsequent application

Specific objectives:
1, 8, 11

Related competencies :
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.

Full-or-part-time: 20h
Theory classes: 4h
Laboratory classes: 4h
Self study: 12h

First chosen module: Concurrency or Parallelism or Distribution

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents associated to the corresponding module and their subsequent application in the practical sessions.

Full-or-part-time: 50h
Theory classes: 10h
Laboratory classes: 10h
Self study: 30h

Second chosen module: Concurrency or Parallelism or Distribution

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents associated to the corresponding module and their subsequent application in the practical sessions.

Full-or-part-time: 50h
Theory classes: 10h
Laboratory classes: 10h
Self study: 30h

Module I: Concurrency

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents associated to Module I (concurrency) and their subsequent application in the practical sessions.

Specific objectives:
8, 9, 10

Related competencies :
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.
CEE2.3. Capability to understand models, problems and mathematical tools to analyze, design and evaluate computer networks and distributed systems.
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.

Module II: Parallelism

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents associated to Module II (parallelism) and their subsequent application in the practical sessions.

Specific objectives:
11, 12, 13

Related competencies :
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.
CTR6. REASONING: Capacity for critical, logical and mathematical reasoning. Capability to solve problems in their area of study. Capacity for abstraction: the capability to create and use models that reflect real situations. Capability to design and implement simple experiments, and analyze and interpret their results. Capacity for analysis, synthesis and evaluation.

Module III: Distribution

Description:
Class preparation with the help of the support material. Understanding and assimilation of the lesson contents associated to Module III (distribution) and their subsequent application in the practical sessions.

Specific objectives:
1, 2, 3, 4, 5, 6, 7

Related competencies :
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.
CEE2.3. Capability to understand models, problems and mathematical tools to analyze, design and evaluate computer networks and distributed systems.
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.
CTR6. REASONING: Capacity for critical, logical and mathematical reasoning. Capability to solve problems in their area of study. Capacity for abstraction: the capability to create and use models that reflect real situations. Capability to design and implement simple experiments, and analyze and interpret their results. Capacity for analysis, synthesis and evaluation.

Exam preparation

Description:
General review and final exam preparation

Specific objectives:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

Related competencies :
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.
CEE2.3. Capability to understand models, problems and mathematical tools to analyze, design and evaluate computer networks and distributed systems.
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.
CTR6. REASONING: Capacity for critical, logical and mathematical reasoning. Capability to solve problems in their area of study. Capacity for abstraction: the capability to create and use models that reflect real situations. Capability to design and implement simple experiments, and analyze and interpret their results. Capacity for analysis, synthesis and evaluation.

Full-or-part-time: 12h
Guided activities: 4h
Self study: 8h

Final exam

Description:
Assimilation of the concepts of course and the exam

Specific objectives:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

Related competencies :
CG5. Capability to apply innovative solutions and make progress in the knowledge to exploit the new paradigms of computing, particularly in distributed environments.
CG1. Capability to apply the scientific method to study and analyse of phenomena and systems in any area of Computer Science, and in the conception, design and implementation of innovative and original solutions.
CEE4.2. Capability to analyze, evaluate, design and optimize software considering the architecture and to propose new optimization techniques.
CEE2.1. Capability to understand models, problems and algorithms related to distributed systems, and to design and evaluate algorithms and systems that process the distribution problems and provide distributed services.
CEE2.3. Capability to understand models, problems and mathematical tools to analyze, design and evaluate computer networks and distributed systems.
CTR3. TEAMWORK: Capacity of being able to work as a team member, either as a regular member or performing directive activities, in order to help the development of projects in a pragmatic manner and with sense of responsibility; capability to take into account the available resources.
CTR6. REASONING: Capacity for critical, logical and mathematical reasoning. Capability to solve problems in their area of study. Capacity for abstraction: the capability to create and use models that reflect real situations. Capability to design and implement simple experiments, and analyze and interpret their results. Capacity for analysis, synthesis and evaluation.

Full-or-part-time: 18h
Guided activities: 2h
Self study: 16h

GRADING SYSTEM

The final grade will be calculated from the grades of the two modules taken by the student as follows NF=0.5*M1+ 0.5*M2

For each module, there will be an exam (EF) and a lab grade (L). The exam will comprise problems on the theory taught. The lab grade will reflect the work done by the students in the practical assignments. The module final grade will be calculated as follows Mi=0.6*EF+ 0.4*L.

BIBLIOGRAPHY

Basic:
- Tanenbaum, A.S.; Steen, M. van. Distributed systems: principles and paradigms. 2nd ed. Pearson Prentice Hall, 2007. ISBN 0136135536.
- Coulouris, G.F.; Dollimore, J.; Kindberg, T.; Blair, G. Distributed systems: concepts and design. 5th ed., int. ed. Addison-Wesley/Pearson Education, 2012. ISBN 0273760599.
- Magee, J.; Kramer, J. Concurrency: state models & Java programs. 2nd ed. John Wiley & Sons, 2006. ISBN 0470093552.
- Armstrong, J. Programming Erlang: software for a concurrent world. 2nd ed. Raleigh, N.C: Pragmatic Bookshelf, 2013. ISBN 9781937785536.
- Goetz, B.; Peierls, T.; Bloch, J.; Bowbeer, J.; Holmes, D.; Lea, D. Java concurrency in practice. Addison-Wesley, 2006. ISBN 0321349601.
- Grama, A.; Karypis, G.; Kumar, V.; Gupta, A. Introduction to parallel computing. 2nd ed. Pearson Education, 2003. ISBN 0201648652.

Complementary:
- Cesarini, F.; Thompson, S. Erlang programming. O'Reilly, 2009. ISBN 9780596518189.
- Herlihy, M. [i 3 més]. The art of multiprocessor programming. Second edition. Cambridge, MA: Morgan Kaufmann Publishers is an imprint of Elsevier, 2021. ISBN 9780124159501.