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.

General:
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.

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.

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.
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 to ensure mutual exclusion, 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 problem of obtaining a consistent global state 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**

<table>
<thead>
<tr>
<th></th>
<th>Theory classes:</th>
<th>Practical classes:</th>
<th>Laboratory classes:</th>
<th>Guided activities:</th>
<th>Self study:</th>
<th>Total learning time: 151h</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>26h</td>
<td>0h</td>
<td>26h</td>
<td>3h</td>
<td>96h</td>
<td>17.22%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17.22%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.99%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>63.58%</td>
</tr>
</tbody>
</table>
Content

Transition systems and process algebra

Degree competences to which the content contributes:
Description:

Understanding parallelism

Degree competences to which the content contributes:
Description:

Concepts underlying distributed systems

Degree competences to which the content contributes:
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, lack of global view, concurrency, lack of a single point of control, security, asynchrony, openness transparency, fault tolerance, scalability. Basic system architectures in distributed systems: centralized (client-server), decentralized (peer-to-peer), hybrid. Types of communication in a distributed system: direct vs. indirect (space and time uncoupling), persistent vs. transient, synchronous vs. asynchronous, discrete vs. continuous. Basic communication paradigms in distributed systems: remote procedure call, message passing, message queuing, group communication, publish/subscribe, shared data spaces, shared-memory, mobile code, stream-oriented.

Safety and liveness properties

Degree competences to which the content contributes:
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.

Degree competences to which the content contributes:
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

Degree competences to which the content contributes:
Description:

Predicting and analyzing performance

Degree competences to which the content contributes:
Description:
Use of models and tools to understand parallelism and performance (Tareador, Extrae, Paraver and Dimemas).

Distributed-memory programming using MPI

Degree competences to which the content contributes:
Description:

Programming GPU devices for computation acceleration using CUDA

Degree competences to which the content contributes:
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

Degree competences to which the content contributes:
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 to ensure mutual exclusion: permission-based algorithms (centralized, Lin's, Maekawa's, Ricart & Agrawala's), token-based algorithms (token ring). 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, the Byzantine generals problem, consensus using failure detectors, Paxos

Distributed shared data: Transactions, consistency, and replication

Degree competences to which the content contributes:

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, two, and three-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, sequential, causal, FIFO. 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)

Qualification system

There will be a final exam (EF) and a lab grade (L). The final exam will comprise problems on the theory taught. The lab grade will reflect the work done by the students in the practical assignments. The final grade will be calculated as follows

\[ NF = 0.6 \times EF + 0.4 \times L. \]
270602 - CPDS - Concurrence, Parallelism and Distributed Systems

Bibliography

**Basic:**


**Complementary:**
