MODULE **DESCRIPTOR**

MODULE TITLE	Parallel Processing and Distributed Systems		
MODULE CODE	EL3008(L6)	**CREDIT VALUE**	20 UK CREDITS / 10 ECTS
SCHOOL	SCHOOL OF SCIENCES		

MODULE **AIMS**

•To convey how parallel programs are used to speed up common computing tasks when deployed on modern multi-core and cluster architectures.
•To convey the fundamental principles of distributed systems on which the Internet and other systems are based on.
•To enhance the students' software development skills by enabling them to construct and evaluate parallel processing and distributed system solutions.
•To enable students to assess relevant parallel processing and distributed systems problems, to choose an appropriate solution, and evaluate it to confirm its fitness to the problem.

MODULE **CONTENT**

Indicative syllabus content:

While parallel processing allows to speedup computations by utilizing appropriate algorithms that execute on multi-core or multi-CPU architectures, distributed systems enable complex scenarios where software components on distant but networked computers cooperate to achieve a task. While in principle a parallel processing system is-a distributed system, the two are rather different as they specialize for a different goal. The module covers two phases, where the first part covers parallel processing solutions, and the second covers fundamentals of distributed systems.

Part A – Parallel Processing
•Parallel hardware and parallel software
•Distributed-memory programming with the Message Passing Interface (MPI)
•Shared-memory programming with Pthreads
•Compiling and running parallel programs on the multi-core and cluster architectures.
•Designing, implementing, debugging, and evaluating the performance of distributed and shared-memory programs.

Part B – Distributed Systems
•Case studies of Distributed Systems: the Web, MMOGs, Web Search
•Challenges in Distributed Systems: Heterogeneity, Openness, Security, Scalability, Fault tolerance, Concurrency, Transparency
•The REST approach as a means for building modern distributed systems
•Time synchronization and Logical clocks
•Voting and Election Algorithms
•Consensus algorithms (reliable networking but unreliable processes)
•Distributed transactions: One-phase and Two-phase commit protocols
•The CAP theorem (and how cloud-computing providers are dealing with it)

INTENDED **LEARNING OUTCOMES**

On successful completion of this module a student will be able to:

1.	Select an appropriate solution for speeding up a task using parallel processing
2.	Choose, implement and evaluate solutions to standard parallelizable tasks such as sorting and searching.
3.	Critically evaluate existing distributed system algorithms and solutions and assess their applicability to open problems.
4.	Implement a distributed application using the REST approach

TEACHING METHODS

A combination of lectures, seminars and practical work will be used:

Lectures introduce concepts, problems and solutions (e.g. basic computer architectures, memory models, hardware parallelization, parallelizable algorithms, challenges of distributed systems, contemporary topics in distributed systems such as global clocks, consensus, leader election, and fault tolerance. Where appropriate, lectures will build on the students' programming skills by illustrating concepts with source code examples.

Seminars will showcase some of the applications of parallel processing and distributed systems, such as their use in speeding up sorting and searching, or for methods used in distributed systems for handling failures. Seminars will also discuss the results of students' investigations, and assist them to prepare for practical work, particularly for programming exercises. Seminars will include reviews of sample code and design exercises. Internet materials will also be used.

Practical work will develop programming and software construction skills related to parallel processing and to specific distributed system concepts. The purpose is to examine and evaluate a variety of methods and technologies rather than to develop a high level of expertise in a specific one. However, to avoid superficiality, as far as possible one programming language and development will be used (e.g. Java and Java Threads, the MPI interface for Java, and cloud-based work using Google Application Engine).

Practical exercises will use a modern IDE—preferred IntelliJ IDEA—to implement examples illustrating parallel processing and distributed system concepts and help the students to apply them to solving practical problems. Examples provided to demonstrate the implementation of concepts using an IDE will be augmented by questions linking the practical work to theory to ensure the students understand the concepts rather than simply follow the recipe.

The examination will expect students to explain, compare, evaluate and apply to simple situations, concepts, tools, languages and techniques relevant to parallel processing and distributed system problems.

The coursework will allow students to develop a prototype, but realistic application and to compare relevant technologies used for speeding up computational problems and for realizing distributed systems. For instance, it will have students implement parallel processing solutions to common problems such as sorting and searching, and evaluate them on parallel or multi-core computers to assess the achieved speedup.

The coursework allows the students to demonstrate practical skills and to integrate techniques explored in the practical labs. They will analyse the concepts, tools and techniques they used in a short evaluative report.

ASSESSMENT METHODS

This module is assessed through a written examination and a coursework.