GHENT UNIVERSITY

Course Specifications

Valid as from the academic year 2026-2027

Parallel and Distributed Software Systems (E017930)

Course size *(nominal values; actual values may depend on programme)*

Credits 6.0 **Study time 180 h**

Course offerings and teaching methods in academic year 2026-2027

A (semester 1)	English	Gent	independent work lecture seminar
B (semester 1)	Dutch	Gent	

Lecturers in academic year 2026-2027

De Turck, Filip	TWO5	lecturer-in-charge
Fostier, Jan	TWO5	co-lecturer

Offered in the following programmes in 2026-2027	crdts	offering
Master of Science in Teaching in Science and Technology(main subject Computer Science)	6	A
Bridging Programme Master of Science in Bioinformatics(main subject Engineering)	6	A
Bridging Programme Master of Science in Computer Science Engineering	6	A
Master of Science in Electrical Engineering (main subject Communication and Information Technology)	6	A
Master of Science in Electromechanical Engineering(main subject Control Engineering and Automation)	6	A
Master of Science in Electromechanical Engineering(main subject Electrical Power Engineering)	6	A
Master of Science in Bioinformatics(main subject Engineering)	6	A
Master of Science in Electromechanical Engineering(main subject Maritime Engineering)	6	A
Master of Science in Electromechanical Engineering(main subject Mechanical Construction)	6	A
Master of Science in Electromechanical Engineering(main subject Mechanical Energy Engineering)	6	A
Master of Science in Computer Science	6	A
Master of Science in Computer Science Engineering	6	B
Master of Science in Computer Science Engineering	6	A
Exchange Programme in Computer Science (master's level)	6	A

Teaching languages

English, Dutch

Keywords

Parallel and distributed software, communication, coordination, synchronization, efficiency, programming models, high-performance computing, major software architectures, fault tolerance.

Position of the course

This course addresses various aspects of the design and implementation of parallel and distributed software. Students receive a state-of-the-art overview of parallel and distributed software systems and learn about different algorithms and software engineering techniques for this important class of software systems. The emphasis is on algorithmic and software aspects, with particular attention to the impact of hardware architectures on performance.

Contents

- Basic concepts, definitions, and classifications.
- Performance metrics and limiting factors: speedup, efficiency, scalability, availability, Amdahl's and Gustafson's laws, CAP theorem, network models, node failures.
- Distributed software: message passing, RPC, RMI, marshalling, message-oriented middleware and services.
- Algorithms for coordination, consensus, and anonymous networks.
- Identification and definition of functional and non-functional requirements for distributed software systems.
- Design of a software architecture that meets the predefined quality requirements using the Attribute-Driven Design methodology and the use of software patterns and design tactics.
- Cloud computing models: service, deployment, and pricing models; cloud platforms and programming models. (for Master of Science in Computer Science students only)
- Microservices and containers (for Master of Science in Computer Science students only)
- High-performance distributed computing: MPI, communication patterns, problem decomposition, case studies.
- Shared-memory programming: data consistency, concurrency, synchronization, deadlocks, thread safety.
- Monitoring and management of large-scale systems: measurements, autonomous management, control-theory-based management.
- Accelerators for artificial intelligence: kernels, thread and memory hierarchy, control flow.

Initial competences

- Basic knowledge of programming in C/C++.
- Basic knowledge of data structures and algorithms.
- Basic knowledge of operating systems.

Final competences

1 Recognize and understand the main algorithmic problems in parallel and distributed software.
2 Know the functions of middleware and the main architectures for parallel and distributed systems.
3 Explain different parallel and distributed software models.
4 Apply basic strategies to solve algorithmic problems.
5 Translate software project requirements into technical and non-technical requirements and design a software architecture that meets these specifications.
6 Develop a basic design for parallel or distributed applications and assess their performance.
7 Take scalability and performance into account when making design decisions.
8 Evaluate and apply algorithms for standard problems.
9 Critically assess alternatives before implementation.

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Seminar, Lecture, Independent work

Extra information on the teaching methods

Theoretical concepts are presented during lectures. In exercise sessions, students apply these concepts in programming exercises and case studies. Independent work includes preparatory tasks and assignments, such as homework exercises.

Study material

Type: Other

Name: Recommended literature and online references
Indicative price: Free or paid by faculty
Optional: no
Additional information: There are no mandatory textbooks with associated costs. Recommended literature and online

references are made available through the electronic learning platform.

References

- George Coulouris et al., Distributed Systems: Concepts and Design (5th ed.), Pearson.
- Rajkumar Buyya et al., Cloud Computing: Principles and Paradigms, Wiley.
- Peter Pacheco, An Introduction to Parallel Programming, Morgan Kaufmann.
- Ian Foster, Designing and Building Parallel Programs, Addison-Wesley.
- Online references: Cloud programming, MPI: The Complete Reference.

Course content-related study coaching

Practical sessions are supervised by teaching assistants. Additional information is provided via the electronic learning platform.

Assessment moments

end-of-term and continuous assessment

Examination methods in case of periodic assessment during the first examination period

Written assessment

Examination methods in case of periodic assessment during the second examination period

Written assessment

Examination methods in case of permanent assessment

Skills test, Participation, Assignment

Possibilities of retake in case of permanent assessment

examination during the second examination period is not possible

Extra information on the examination methods

Periodic evaluation consists of a closed-book written exam.
Non-periodic evaluation consists of the assessment of homework assignments.

Calculation of the examination mark

75% written exam
25% homework assignments

If the exam score is below 8/20, the final score cannot exceed 8/20. In the second examination period, scores for practicals and assignments are only included if they improve the final grade.