Carleton UNIVERSITY | Department of Systems and Computer Engineering

SYSC 4102
Performance Engineering

Calendar description

Techniques based on measurements and models, for predicting and evaluating the performance of computer systems. Instrumentation. Simple queueing models and approximations. Techniques for modifying software designs to improve performance.

Includes: Experiential Learning Activity.

Lectures three hours a week, laboratory/problem analysis three hours alternate weeks.

http://calendar.carleton.ca/undergrad/courses/SYSC/

Prerequisites

(ECOR 2050 or STAT 3502) and SYSC 4001.

Also offered at the graduate level, with different requirements, as SYSC 5101, for which additional credit is precluded.

Prior knowledge

This course involves the analysis of software. Students are expected to know how to program in a systems programming language such as C, and understand how operating systems work, in particular inter-process communication and synchronization.

Course objectives

Performance in this course deals with time and capacity characteristics of computer-based systems. When systems are designed for functionality only, the time to complete a response may be so long that the system is ineffective; similarly, its capacity to serve a large number of users may be inadequate making the system uneconomic to use. Both of these problems occur often in practice. Performance engineering that aims at solving these problems is a body of concepts and techniques for evaluating systems and system designs, using measurements and models.

Meeting performance requirements (such as response time, throughput, etc.) is a major concern for all kinds of software products with performance constraints, and especially for real-time systems. Software Performance Engineering (SPE) addresses performance issues throughout the whole software lifecycle and aims to ensure that software products under development will meet their performance requirements. SPE uses predictive performance models to assess different design alternatives at an early stage, before major obstacles to performance are frozen in design and code. This can

improve the quality of the final product by helping designers to make informed choices and trade-offs early in the life cycle, when changes are not expensive and open alternatives still exist. As the product evolves, so does the performance model, capturing more system features and producing more accurate results.

The course will cover different basic approaches to performance engineering. Topics will be chosen from measurement techniques, interpreting and comparing results, models that explain capacity constraints and delays (bottleneck models, queueing models and layered queueing models), an introduction to performance-oriented design based on performance principles, patterns and antipatterns.

The goal of this course is to prepare the students to address performance problems in real-time concurrent and distributed systems, such as embedded controllers, enterprise distributed systems, web services-based systems and cloud systems. It will introduce the conceptual framework and the nature of performance problems and solutions, so that the student can apply them into the field.

List of topics

- Performance concepts and requirements
- Performance measurement. Workloads
- Performance models. Cures for performance problems
- Memory hierarchy effects
- Queueing Analysis
- Software resources
- Layered resource effects
- Measurement and tools
- Software Performance Engineering
- Software execution models and system execution models
- Performance Oriented Design: performance principles, patterns and anti-patterns

Learning outcomes

By the end of this course, students should be able to:

- Compute performance bounds for Queuing Network Models.
- Apply fundamental operational laws for computing performance measures.
- Apply QN and LQN models for analyzing and improving software performance.
- Understand the meaning of system bottleneck and techniques to alleviate it.
- Employ tools to measure the performance of software.
- Apply performance principles, performance patterns and antipatterns to improve software performance.

Graduate Attributes (GAs)

The Canadian Engineering Accreditation Board requires graduates of engineering programs to possess 12 attributes at the time of graduation. Activities related to the learning outcomes listed above are measured throughout the course and are part of the department's continual improvement process. Graduate attribute measurements will not be taken into consideration in determining a student's grade in the course. For more information, please visit: https://engineerscanada.ca/.

Graduate Attribute	Learning outcome(s)
1.8.S - Knowledge base: Discipline-specific concept SCE-5: Software Engineering	
5.3 - Use of engineering tools: Tools for design, experimentation, simulation, visualization, and analysis	

Instructor and TA contact

Specific to course offering (tbd)

Textbook (or other resources)

Specific to course offering (tbd)

Evaluation and grading scheme

Specific to course offering (tbd)

Breakdown of course requirements

Specific to course offering (tbd)

Tentative week-by-week breakdown

Specific to course offering (tbd)

General regulations

Specific to course offering (tbd)