inf | Informatik
Computer Science

ETH

Advanced Systems Lab

263-0007-00L

D-INFK, Fall Semester 2015

Project and Course Description

~

Gustavo Alonso

Systems@ETH Zürich

Course Structure

This is a project based course operating on a self-study principle. The course relies on the student's own initiative, aside from a few tutorials discussing the project and some basic supporting material, there will be no lectures in the conventional sense. The project as well as the course requires that the student complements the material covered in the tutorials with his/her own study as not all relevant and/or needed material will be covered. The grading will be based on the project; there is no exam.

The pre-requisites for this course include knowledge at the Bachelors Level of the following topics:

- Database programming (schemas, SQL, indexing, installing and operating a database engine)
- Programming in Java (general knowledge of OO programming, threading, concurrency, and debugging)
- Networks (understanding of how computer networks operate and the basis of performance evaluation in networks)
- System Design (operating systems, concurrent systems, low level optimizations, instrumentation)
- Statistics
- Algorithms and data structures

These topics will not be covered during the course but knowledge of them is assumed. If a student lacks the necessary background, it is expected that he/she will acquire the necessary knowledge on his/her own. No course or project requirements will be changed for individuals because of lack of previous knowledge in any of these areas. Students who do not have the necessary background or system development skills may want to consider taking instead one of the other two Advanced Labs offered by the department to comply with the requirements for the masters degree in computer science.

Course Objectives

The main goal of this course is to learn how to evaluate the performance of complex computer and software systems. The course offers an opportunity to bring together the knowledge and expertise acquired in different areas (networking, databases, systems programming, parallel programming) by allowing students to build a multi-tier, distributed information system. The course focuses less on system design and development and more on the evaluation and modeling of the system: understanding the behavior, modeling its performance, code instrumentation, bottleneck detection, performance optimizations, as well as analytical and statistical modeling. The ability to explain the behavior of the system plays a bigger role in the grading than the actual building of the system. However, we expect designs and code that have been thought through – major design mistakes and bad coding practices will affect the grade.

Course Evaluation

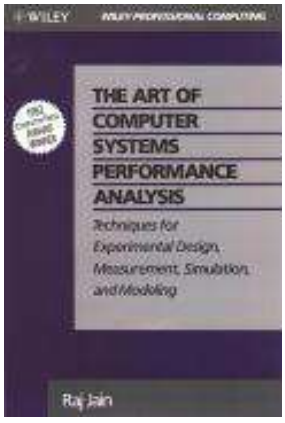
The course will be evaluated through a project. The project consists of two milestones, the first covers the system design, implementation, and experimental evaluation; the second covers the analytical evaluation of the system and the experimental results. All milestones need to be delivered to obtain a passing grade in the project. Failing the project implies failing the course. Work in the project will be done on an individual basis. The milestones include delivering system code, a report, and experimental data. Students might be required to present the results in person and explain the system or results in the report. The dates for submitting the deliverables due at each milestone are fixed and cannot be changed. Failure to submit the results on time will result in failing the project (the submission system –a repository that will be presented in the exercise session- will automatically close at the indicated date and time). The deadlines for the two milestones are as follows:

- 6th November 2015, 18:00
- 18th December 2015, 18:00

Each milestone is worth 300 points and both must be completed to pass the course. It is important to note that the evaluation of the project is not based on completing a number of clearly specified task. The goal is to build a system *and explain its behavior*. Developing a system that works but for which no explanation can be produced on why it behaves the way it does is not sufficient to reach a passing grade. Similarly, collecting experimental data or developing a model is not enough to pass the corresponding milestones. A passing grade is reached by having the proper explanations for the data and the model, including its potential discrepancies with the implemented system.

Supporting Material

The following text book (available in the library) can b very useful for the course: *The Art of Computer Systems Performance Analysis, Raj Jain, Wiley Professional Computing, 1991*. Of particular relevance are the following chapters:



Chapters 1, 2, 3 = general introduction, common terminology
Chapters 4, 5, 6 = workloads
Chapter 10 = data presentation
Chapters 12, 13, 14 = probability and statistics
Chapters 16, 17, 18, 20, 21, 22 = experiment design
Chapters 30, 31, 32, 33, 36 = queuing theory

Project

The project consists of the (1) design and development of a message based middleware platform as well as the evaluation and analysis of its performance, and (2) development of an analytical model of the system and an analysis of how the model predictions compare to the actual system behavior.

General Remarks

This course is a demanding one not because the task at hand is in itself difficult but because for many students it is the first time they are confronted with designing a system with several degrees of freedom and where skills from several areas of computer science need to be brought to bear into the problem to solve it. *Waiting until close to the deadline to complete a milestone will not work on this course*. Starting as early as possible is essential, and aiming at completing it well before the deadline is highly recommended so that there is enough time to analyze and understand what has been built. In this course, the focus is on understanding the system and being able to explain what it does and why it behaves the ways it does; this is more important than development of the system itself. No matter how much effort goes into the development, if the system is ready only shortly before the deadline, there will be no time for a complete experimental analysis and for developing a thorough understanding of the system's performance characteristics. However, it is mainly on these latter aspects where the project will be evaluated.

Experience from last editions of the course show that there are a number of typical problems with the project that occur because of poor time management:

- System works but behavior of performance traces cannot be explained (why response time grows over time?, why throughput peaks for a given number of clients?, what are the bottlenecks in the system?)
- System works but the traces are incomplete, experiments have not been repeated a sufficient number of times to achieve statistical significance, lack of proper statistical treatment of the data (confidence levels, deviation, factor analysis)
- Inconsistent results in the data collection with different experiments showing contradictory behavior
- Poor modeling or no explanations for the discrepancies between the model and the system
- Unexplained behavior of the system from a performance perspective

Even if the system is complete and works well, these situations typically lead to failing the milestones so make sure you devote sufficient time and effort to the analysis of the system.

In this project, a number of well established professional practices are highly advisable: create a timetable for your project and adhere to it, monitor your progress so that potential delays can be identified and dealt with early enough, use a version control system for development (will be provided), heavily document all your code, plan your experiments and experimental work, use a database to store your experimental data, keep a journal of experiments and experimental parameters, keep track of result files and data to make the analysis simpler later on. Make sure you have backup copies of your code, data, and reports.

Infrastructure

The project requires a more extensive infrastructure than probably most students have ever used for a course project. A local computer cluster is available for this course where students can develop the system and do some basic testing. For testing and measurements, an account will be provided in the cloud computing infrastructure of Amazon for conducting the development and performance analysis of the system. Students will get a coupon that should cover all the time necessary for completion of the project.

Communication

Please regularly check the web page of the course. Also, make sure that your teaching assistant is informed of the progress you have made in the milestones. Avoid raising major issues in completing a milestone shortly before the deadline. An e-mail address is available for questions, requests, and feedback regarding the course: sg-asl@lists.inf.ethz.ch.

Milestones, Reports, and Presentations

The deliverables for each milestone are as follows:

	Milestone 1	Milestone 2
System code	Code, Scripts for experiments	
Experimental data	Basic tests and simple traces, Long running traces, summary of raw data and graphs for all experiments (max 1 Gbyte!!)	
Written report	Architectural diagrams, interface descriptions, explanation of system design Description experimental infrastructure, description of all experiments, statistical treatment of data, graphs, commentary and analysis	Model and analysis of the system, additional experiments (if any), commentary and analysis
Oral presentations	On demand after the deadline (to clarify questions from our side)	On demand in case there are questions from our side, presentations on the second/third week of January

Milestone 1 – Part 1

The goal is to develop a message passing middleware platform supporting persistent queues and a simple message format. The idea is to build a *simpler* version of systems such as Apache ActiveMQ, IBM MQSeries, or JBoss Messaging. The programming language to use is Java and the database underlying the system should be PostgreSQL.

The only external libraries allowed in Java is the PostgreSQL JDBC driver and log4j. Out of the JDK you are not allowed to use the JPA (Java Persistency API) nor the RMI (Remote Method Invocation).

System Architecture

From the application's perspective, messages are strings sent to a queue. Internally, a message contains more information (needed to handle it appropriately). The recommended attributes for a message include (you may decide to add more depending on the functionality implemented):

Message identifier: an integer number uniquely identifying the message
Sender: an integer number uniquely identifying the sender
Receiver: an integer number uniquely identifying the receiver
Queue: an integer number uniquely identifying the queue
Time of arrival: a timestamp indicating when a message arrived at the queuing system
Message: the string containing the actual message (consider two lengths 200 and 2000 characters)

The system should be built in a distributed manner with three different tiers connected through well defined interfaces. Each tier has to be built in such a manner that it can run on a separate computer from the other tiers. However, some of the initial correctness tests can probably be done in a single machine with each component running in its own process.

The lowest tier implements persistent queues by using a database engine (PostgreSQL). The second tier (messaging system) implements the queuing system proper and is in charge of message manipulation and the system logic related to message management. There should be at least two such modules working in parallel and on different machines. The third tier is implemented as part of the clients that send and receive messages. The basic functionality the system must support is as follows:

- Messaging
    - Clients can create and delete queues
    - Clients can send and receive messages
    - Clients can read a queue by either removing the topmost message or just by looking at its contents
    - Clients can send a message to a queue indicating a particular receiver
    - If a message has an explicit receiver, it can only be accessed by that receiver (but there is no need to implement a client authentication system)
    - Clients can query for messages from a particular sender (at most one message is returned)
    - Clients can query for queues where messages for them are waiting
- Management
    - Clients use fixed accounts in the messaging system and are uniquely identified by these accounts
    - All clients can send and receive from any queue (no access control in the system)
    - Sending or reading from non-existing queues, reading from an empty queue, or failing to create or delete new queues are events that must raise well defined errors
- Persistence
    - Clients, queues, and messages are stored persistently in a database (lowest tier). This information must survive system failures

    - Queues and messages are indexed
- Deployment
    - There is only one database
    - The second tier is two or more independent (but identical) messaging components
    - Clients connect to a messaging component of the second tier

Design

Database: The design involves coming up with an adequate database schema for the messages and queues, the corresponding indexes, as well as the necessary stored procedures for sending, receiving, and manipulating queues and messages. Messages, clients, and queues must have unique, system wide identifiers that are stored persistently in the database. Developing a simple management console should that displays the status of the system: number of queues created, messages in each queue, etc. is an useful tool (but not mandatory). In the machine running the database, the only allowed process is PostgreSQL (no proxies, no additional tiers, etc.).

Messaging system: This is a Java application that can be run concurrently on different machines. Each instance must be capable of:

- supporting at least 30 concurrent clients
- using multi-threading internally
- maintaining a connection pool to the lower tier
- logging its activities for analysis (messages sent, received, relevant events, etc.)

Clients: Clients should include instrumentation to measure elapsed time between sending and receiving as well as logging capabilities to keep track of messages sent and received.

Milestone 1 – Part 2

The goal is to experimentally evaluate the system built in milestone 1, part 1 to determine its performance characteristics. We are interested in the throughput and response time that can be sustained as a function of different system parameters. We are also interested in what parts of the system or which system components contribute to the observed throughput and response times, as well as identification of the potential bottlenecks in the system. The system has to compile and run using ANT (a tutorial will be given).

Questions to answer

- Traces
    - Run the system for 30 minutes under well defined load and measure key performance parameters, observing the overall behavior.
- Micro-benchmarks:
    - How long does it take to send a message?
    - How long does it take to receive a message?
    - How much time is spent for each of those operations in each part of the system? Consider the client, the messaging component, the database, and the network links between them

    - Find the limits of each component of the system for your particular deployment and hardware configuration (how many messages can a client send per second?, what is the think time caused by the code?, how many clients/connections can a messaging component handle?, how does the messaging system behave as a function of the number of connections to the database?, how many messages can the network support?, etc.)
- System behavior
    - Identify the parameters that affect the performance (throughput, response time) of the system: number of messages in the dataset, number of clients, number of messaging components, size of the messages, sending rates, etc. For each such parameter, evaluate how throughput and response time behave.
    - Explore the scalability of the system as more instances of the messaging system are added. Can you find the limit? What is the bottleneck?
- Characterizing the code
    - What are the bottlenecks in your code?
    - What are the operations in the code that are most expensive and determine the overall performance characteristics?
    - What are the data structures that play the most critical role in the behavior of the system?

Techniques

The evaluation must be done through complete sets of experiments conducted in a controllable and reproducible manner. The statistical significant of the data and the results must be clearly explained (repeat the experiments for as long as needed to get convincing confidence intervals). Define in advance the factors and parameters to explore as well as the measurements to take. Use the methodology presented in class for 2^k factorial designs to ascertain the influence of different factors in the behavior of the system. Also use the methodology explained in the text book (chapter 23) for identifying the configurations with the best performance.

Milestone 2

The goal of the final milestone is to develop an analytical queuing model for each component of the system and for the system as a whole. Using the model, derive the performance characteristics that the model predicts and compare them with the results obtained in milestone 1. Explain where the model and data match and where they do not match, including sufficiently detailed explanations of the code/system/hardware characteristics behind the observed behavior.

The description of the modeling effort should include the queuing model(s) and parameters used for each component. The overall system should be modeled as queuing network and the experimental results analyzed using the operational laws. In the report clearly indicate the behavior expected from the model through graphs and plot them together with the measured behavior. When evaluating the experimental data and the models, clearly indicate the laws you are applying and explain why you think they can be applied in the corresponding analysis.