Advanced Systems Lab

263-0007-00L

D-INFK, Fall Semester 2012

Project and Course Description

~

Gustavo Alonso – Donald Kossmann

Course Structure

This is a project based course operating on a self-study principle. The course relies on the student's own initiative rather than on covering every single detail of everything needed for the course. The course will include several supporting lectures and exercise sessions to introduce the project and to provide the basic knowledge for the milestones to be completed. There will be 7 or 8 such lectures at the beginning of the semester. The project as well as the course requires that the student complements the material covered in the lecture with his/her own study as not all relevant material will be covered.

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.

Lecture: Tuesday, 17:15 – 19:00, CAB G 61.
Exercise sessions: Thursday 17:15 – 19:00, CAB G 52, CAB G 56, CHN D 42, CHN D46, CHN D 48.
Project (3 milestones): during the semester
Exam: Written exam during the exam session (Jan/Feb 2013)

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 courses by allowing students to build a multi-tier, distributed information system involving databases, networking, concurrent programming, and distributed systems. The course focuses not only on system design and development but also on the evaluation and modeling of systems: understanding their behavior, modeling their performance, code instrumentation, bottleneck detection, performance optimizations, and analytical and statistical modeling. The course places as much emphasis on design as on evaluation/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 and an exam. The project weights 40% of the grade and the exam 60%. All milestones need to be delivered to obtain a passing grade in the project. Both project and exam must be passed independently of each other to pass the course.

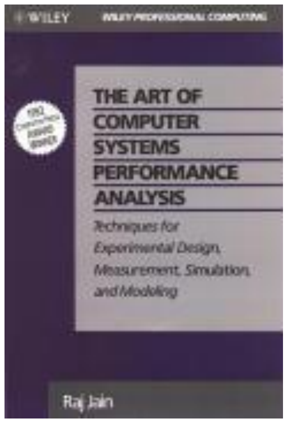
The project will be completed in three milestones during the semester. Two of the milestones will be done in teams, the final milestone is an individual one. The milestones include delivering system code, a report, experimental data, and giving a presentation about the results achieved. The dates for submitting the deliverables due at each milestone are fixed and cannot be changed. Failure to submit the results on time will result in the milestone graded with zero points (the submission system –a svn repository that will be presented in the exercise session- will automatically close at the indicated data and time). The deadlines for the milestones are as follows:

- 18th October 2012, at midnight
- 14th November 2012, midnight
- 19th December 2012, midnight

The exam is a written exam during the exam session. It includes questions about the material covered in the lecture (modeling and analysis) as well as questions on the project and the milestones.

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.

Exam Material


WILEY
WILEY PROFESSIONAL COMPUTING
THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS
Techniques for Experimental Design, Measurement, Simulation, and Modeling
Raj Jain

The course will be based on the following text book: *The Art of Computer Systems Performance Analysis, Raj Jain, Wiley Professional Computing, 1991*. Of this textbook, the exam will be based on 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

We do not expect students to memorize the expressions of, e.g., queuing theory models. We expect students, however, to be able to derive the basic expressions for the simple queuing models (M/M/1, M/M/m, M/M/1/B, etc.) and to be able to work with the different operational laws to solve modeling problems. Note that queuing networks is included in the material (and are needed for milestone 3).

Note that the material in some of the chapters listed may require to read other chapters not listed as part of the exam. These chapters must also be read, they are not included in the list as there will be no direct questions about them in the exam.

Project

The project consists of the (1) design and development of a message based middleware platform, (2) evaluation and analysis of its performance, and (3) 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 many 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* so make sure you start as soon as possible working on the project and aim at completing it well before the deadline so that there is enough time to analyze and understand what has been built. In this course, understanding the system and being able to explain what it does and why it behaves the ways it does is as important (more important, in fact, for the last two milestones) as the 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 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 computer cluster is available for this course where students can complete the project. However, in the past, the management of the cluster has proven to be a complex task due to undisciplined use (not respecting reservation times), unprofessional behavior (not cleaning up behind you), and too many people wanting access to the cluster shortly before the deadline. These situations lead to a great deal of frustration on all sides. This year, to be able to better enforce rules and ensure appropriate behavior on the cluster, reservation slots will be pre-allocated at random to each group. No access will be permitted beyond the allocated slot. Make sure that you arrange your work on the project to make the most of the slots available to your group.

As an alternative or complement to using the cluster, we strongly suggest students to consider using a cloud computing infrastructure such as that of Amazon for conducting the development and performance analysis of the system. The cost for the work to be done in the course is low –comparable to that of buying scripts and other material for the courses- and a cloud account provides a greater deal of freedom than what can be guaranteed with the course cluster. The assistants for the course can help with the setup of accounts and general advice although these systems should be readily usable to any student of computer science.

Teams

At the beginning of the course teams of 3 people will be formed. Students are not allowed to work individually in the first two milestones – the project complexity makes it very difficult to accomplish the necessary results by one person. The first two milestones of the projects are to be done in teams and the project results will be graded as a team. The last milestone is to be done individually. Problems within teams are to be reported to the corresponding assistant immediately so that they can be corrected in time.

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	Milestone 3
System code	Code	Scripts for experiments	
Experimental data	Basic tests and simple traces	Long running traces, Raw data and graphs for all experiments	
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 presentation (~10 minutes)	Database schema and interface design on Sept. 27th As a team in exercise session on 18th October	As a team in exercise session on November 15th	Deadline 20th December, if there are questions from our side presentation on the first week after semester

Milestone 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 library that you are allowed to use in Java is the PostgreSQL JDBC driver. 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 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 sender
Sender: an integer number uniquely identifying the sender
Receiver: an integer number uniquely identifying the receiver (optional)
Queue: an integer number uniquely identifying queue
Context: an integer identifying an application context for a given message

Priority: an integer number between 1 and 10 indicating the priority of the message (10 highest, 1 lowest)
Time of arrival: a timestamp indicating when a message arrived at the queuing system
Message: the string containing the actual message (max. size 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. 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
    - Messages can be one way or request-response interactions
    - One way messages have no context
    - Request-response interactions are identified by the sender and context (request) and the context and receiver (response)
    - Clients can create and delete queues
    - Clients can send and receive messages
    - Clients can read a queue without removing messages
    - Clients can retrieve a message by removing it from the queue
    - When retrieving a message, clients can request to retrieve the earliest one or the one with the highest priority. By default, retrieving a message from a queue returns the one with the highest priority
    - Clients can send messages to multiple queues
    - 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
    - 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 exceptions
    - A management console directly connected to the lowest tier displays the state of all queues and messages in the system (better implemented as a GUI but not necessary)

- Persistence
    - Clients, queues, and messages are stored persistently in a database (lowest tier). This information must survive system failures.
    - Queues and messages are indexed (only the relevant attributes)
- Deployment
    - There is only one database
    - The second tier can be one 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. Examples of the functions the database should support include:

- Create_Sender
- Create_Receiver
- Create_Queue (Name)
- Delete_Queue (Name)
- List_queues
- Put (message, queue)
- Get (queue)
- Read (queue)

Messages, clients, and queues must have unique, system wide identifiers that are stored persistently in the database. A management console should be developed that displays the status of the system: number of queues created, messages in each queue, 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
- caching messages to enhance performance
- logging its activities for analysis (messages sent, received, relevant events, etc.)

Clients: There are two types of clients. Clients that communicate through one way messages and clients that interact through request-response messages. Clients should include instrumentation to measure elapsed time between sending and receiving as well as logging capabilities to keep track of message sent and received.

Basic testing

After the first week, the groups will do a presentation in the exercise session of the schema and interface designed. The system should be tested sufficiently to show that it works and has reasonable performance. We suggest running simple traces and checking that all interfaces work under a number of different configurations. The report should explain how the system has been tested and any problems encountered. You should be ready to do a demo of the system if requested to do so.

Milestone 2

The goal is to experimentally evaluate the system built in milestone 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 through and response times, as well as identification of the potential bottlenecks in the system.

Questions to answer

- Traces
  - Run the system for 2 hours under a moderate to high 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 (throughout, 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

For the traces, use at least 50 client processes using one way messages and 30 clients doing request-response interactions. The one-way clients send a message at random to one of the other clients and wait for a message for them. Upon receiving one message, they mark it and send it again to another random client. Initially the messages can be a counter set to 0 that is incremented in each iteration. That way the final messages contain how many times they have been exchanged. The message could also contain the identifiers of the clients involved to trace back all the exchanges.

Half of the request-response clients act in pairs with one acting as client and the other as server (one makes requests and the other answers them). Use counters and information in the messages exchanged to keep track of what is going on.

The other half of the request-response clients should act as a set of clients requesting a service (by sending messages to a queue) and a pool of servers providing the service by retrieving requests from the pool.

The system should be run uninterrupted for (at least) 2 hours. Obtain traces of the number of messages exchanged over time for each type of interaction, the average throughput throughout the experiment, and the observed response time of the system (as round trip times). Make sure you check whether messages are being lost and, if so, find out where.

The evaluation must be done through complete sets of experiments conducted in 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 book (chapter 23) for identifying the configurations with the best performance.

Milestone 3

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 2. 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.