Advanced Systems Lab (Intro and Administration)

G. Alonso

Systems Group

http://www.systems.ethz.ch

Overview of the Course

- Focus on project
  - Individual project during semester (3 milestones)
- This is a **project based** course not a lecture:
  - Organized around tutorials
  - Self-studying
  - Learning by doing

Objective of this course

- Quantitative evaluation of computer systems
- Basics of performance evaluation
- Basics of queuing theory
- Learn how to answer questions of the form...
  - What is better: A or B?, Is A good enough?
  - What are the limits of A?, How can I improve A?
- A, B, C, ... are computer systems
  - software + hardware
  - often only components of a bigger system

Why?

- The world has changed:
  - Cloud computing: services instead of programs
  - Data centers: economies of scale
  - Cost of IT: computing is now an important part of the world's energy budget
- Performance, scalability, reliability, energy consumption, resources needed, etc. are becoming key design constraints for software

From a former student …

… coming back to work in the inter-semester break, the first project assignment I got was to analyze and improve the performance of the trading systems and prototype a new architecture … . It was an unbelievable feeling to leave the exam room and get such work assignment that requires all the knowledge I learned in class.

Goals

- Things you will know at the end of the course
  - System building principles for cloud computing
  - Experimental design and analysis
  - Answering quantitative questions about systems
  - Experiments presentation
  - Queuing theory applied to system performance analysis
  - Insights on the systems used in the project

Communication

- Web page

  http://www.systems.ethz.ch/courses/fall2016/asl

- E-mail list (not public): sg-asl@lists.inf.ethz.ch
- Assistants:

  Claude Barthels

  David Sidler

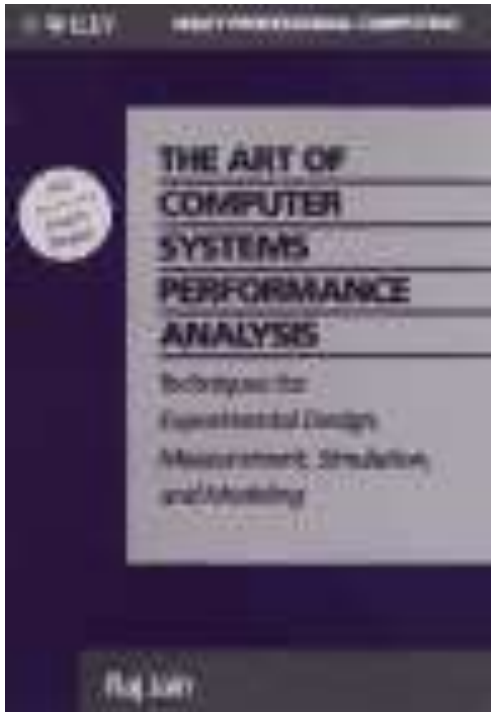
  Kaan Kara

  Mohsen Ewaida

  Zsolt Istvan

Recommended reading

- Raj Jain: The Art of Computer Systems Performance Analysis, John Wiley &Sons (Version 2)
  - (we will not cover Part V „Simulation“)
  - Library has copies. Nevertheless, worth buying.
  - Crucial reading for project



Tutorials

- Examples, basic guidance, and overview of needed tools through tutorials
- I - System Design and messaging systems
- II – Experimental design and statistics
- III – Queuing theory
- IV – Examples of applying queuing theory
- Azure cloud tutorial + Project Tutorial + Tools tutorials

Help with the project

- Regular Q&A sessions with assistants
  - Send questions in advance
  - Discuss your project
  - Get feedback
  - Learn from others
- You need to make your own design decisions, please do not ask us to make them for you and do not demand we validate them. That is your job!

Bootstrapping the course

- Today = Introduction
- This coming Thursday = project tutorial (G 61)
- Tuesday next week = Tutorial
- Thursday next week = Azure tutorial (G 61)

THE PROJECT

Requirements

- Things we assume you know:
  - Programming (Java)
  - Basic statistics and probability theory
  - Operating systems (threads, scheduling, memory management)
  - Databases (SQL)
  - Networks (RPC, TCP/IP, routing, sockets)
- If you do not have the necessary background, you need to acquire it before taking this course

Build a system ***and*** explain it

- The project is to build a small key value store similar to those used in cloud platforms
- Miniature version of real ones
- The goal is to:
  - Build it (milestone 1)
  - Measure it (milestone 2)
  - Explain it (milestone 2)
  - Analyze it (milestone 3)

Project for this course

- Individual project
- Three milestones (200 points each, total 600). You need 400 points to pass and enough points on each milestone
- Milestone 1 (200 points)
  - Part 1: Build System
  - Part 2: Baseline and basic trace
- Milestone 2 (200 points)
  - Part 1: performance experiments
  - Part 2: performance analysis (explain results)
- Milestone 3 (200 points)
  - Part 1: formal analysis
  - Part 2: explaining results and system

“How to” suggestions

- Design architecture, produce design document
  - Discuss design with assistant at Q&A session
- Incremental and iterative development
  - Always have a working system, then add to it
  - Continuous testing
- Detailed time plan (and stick to it)
  - Development, deployment, testing
  - Experiments
  - Data processing
  - Report writing

“How to” suggestions

- Start early (experiments take longer than you think and bring surprises)
- There is enough time in the semester to complete the project successfully. But you will not be able to do it if you try to complete it in the last minute.

A word of caution

- The project is in itself not difficult
- It can, however, become very, very difficult if:
  - You do not allocate enough time
  - You do not know what you are programming
  - You make things more complex than necessary
  - You do not understand what you want to do with the experiments
- We are not after perfect systems but about the ability to explain what has been done in a professional and mature way.

THE SYSTEM

A database engine

- Data stored following a schema
  - Tables, constraints
  - Smaller addressable unit: tuple
- Data accessed and manipulated through SQL
  - Relational algebra, well defined execution model
  - Well defined operators, optimizer
- ACID through transactions
  - Maintain data consistency
  - Ensures recoverability and persistence

Schemas

- Organize the data
- Through normal forms, avoid redundancy and structure the data in clear units (tables)
- Through constraints guarantee data integrity

- Must be done before data can be used in a database
- Dominant decomposition a problem (views help but only so much)

SQL

- Well defined interface
- Efficient implementations
- Declarative language

- Only limited set of operators
- Impedance mismatch between SQL and regular programming languages

ACID

- Strong guarantees
- Work done by the database engine, not the application
- Well defined properties and understandable concurrency

- Expensive, both in throughput and response time
- Does not scale that well

Get rid of everything

- No schema
  - Just a key and blob attached to it, whose structure is unknown
- No SQL
  - Get and Set as the only operations
- No ACID
  - Eventual consistency
  - Potentially no transactions

= Key Value Store

Key Value Stores

- Schema:
  - Key + BLOB (or pointer to blob)
  - Index on key (hashing)
- Deployment
  - Horizontal partitioning
  - Replication of partitions
- Replication strategy
  - Quorums (simple majority)
  - Asynchronous replication across all copies (eventual consistency)

KVS: advantages

- Very fast lookups (hashing)
- Easy to scale to many machines (simply add more copies/machines)
- Useful in many applications:
  - Lookup user profiles
  - Retrieve users web pages and info
  - Scalable and easy to get going (no schema)

Many variations

- Cloud computing has given raise to many variations on this theme
  - Most serious system converging back towards a full blown transactional engine with a schema and SQL support
  - Specialized systems used as accelerators (memcached) or concrete application stages (profile look up)

THE MEASUREMENTS

Characterize the performance of the system you have built

- Throughput, response time, scale-up, scale out, bottleneck analysis, 2K analysis, imapct of design features, measuring relevant factors …
- More information about this in the tutorials and the book
- Plan your experiments
- Give yourself enough time to run experiments
- Give yourself enough time to understand the data

THE ANALYSIS

Modeling

- Analyze the system using interactive law
- Build different queuing models, solve them analitically, compare with actual measurements, explain differences, characterize your system
- More information in tutorials and book