Advanced Systems Lab (Intro and Administration)

G. Alonso

Systems Group

http://www.systems.ethz.ch

THE COURSE IN A NUTSHELL

Objectives of the 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 course

- 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
- Performance no longer comes automatically from hardware upgrades

Side effects

- Learn about processing data at high rates
- Learn about code instrumentation and performance measurement
- Learn about complex, distributed systems
- Learn about cloud computing

- The code and use cases are real and correspond to systems used in practice (this is an advantage and a disadvantage)
- This is a demanding but fun project

Results

- 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
  - Explaining and reporting on your work
  - Queuing theory applied to system performance analysis
  - Insights on the systems used in the project

BASIC ADMINISTRATIVE INFORMATION

Overview of the Course

- Focus on project
  - Individual project during semester
    - Build a small system
    - Evaluate and model its performance
    - Explain the results
  - Final report (plus code and data) due on last week of the semester
  - Project description and report template will be provided
  - No exam
- This is a **project based** course not a lecture:
  - Organized around tutorials and examples
  - Self-studying
  - Learning by doing
  - Requires you to be pro-active

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.
  - <u>Crucial reading for project</u>

Communication

- Web page
  https://www.systems.ethz.ch/courses/fall2017/asl
- E-mail list (not public): sg-asl@lists.inf.ethz.ch
- Assistants:
  Claude Barthels
  David Sidler
  Kaan Kara
  Mohsen Ewaida
  Zsolt Istvan

Tutorials and Exercises

- Tutorials:
  - Help with the conceptual aspects of experimental design, data gathering, performance analysis, and reporting
- Exercises
  - Help with tools and environment
  - Detailed description of the project and work to do
  - Examples of what needs to be done based on similar systems (but not the one you have to build)
- Will proceed until beginning/mid November

On attendance

- It is highly recommended that you attend the tutorials and the exercises. They are there to help you with the project.
- Even if you are a good systems programmer, it is unlikely you know enough about performance analysis
- Initial exercises will be in a single room (explaining the project, explaining the cloud platform, explaining the basic tools), then in smaller groups for actual exercises and examples (see web page)

Important information

- This is one of three mandatory courses for all master students in CS. You need to pass two out of these three courses.
- It is strongly advised to pass the two necessary courses in year 1 of the master
- Unless you really know what you are doing, it is not advisable to take ASL and the Algorithms Lab at the same time (much less if you are also taking other courses involving projects)
- **DEADLINE FOR DEREGISTERING**: October 15th

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 ENVIRONMENT

Working on the cloud

- Development can be done on a local machine (laptop, desktop)
- System and all experiments must run on the cloud (MS Azure)
  - Distributed
  - Using VMs
- You will get a sufficient amount of free credit (treat it carefully)
- Important to know about the cloud but it was drawbacks …

Experiments in cloud computing

- The basic premise of cloud computing is that you are sharing the infrastructure with potentially very many concurrent users
- Does not affect functionality but it might affect performance
- This course is about studying and understanding performance, the cloud adds an additional challenge to understanding what is going on
- The problem is easy to solve if you think about it

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 system around a key value store similar to those used in cloud platforms
- Miniature version of real ones
- The goal is to:
  - Build it
  - Measure it
  - Explain it
  - Analyze it

“How to” suggestions

- 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.

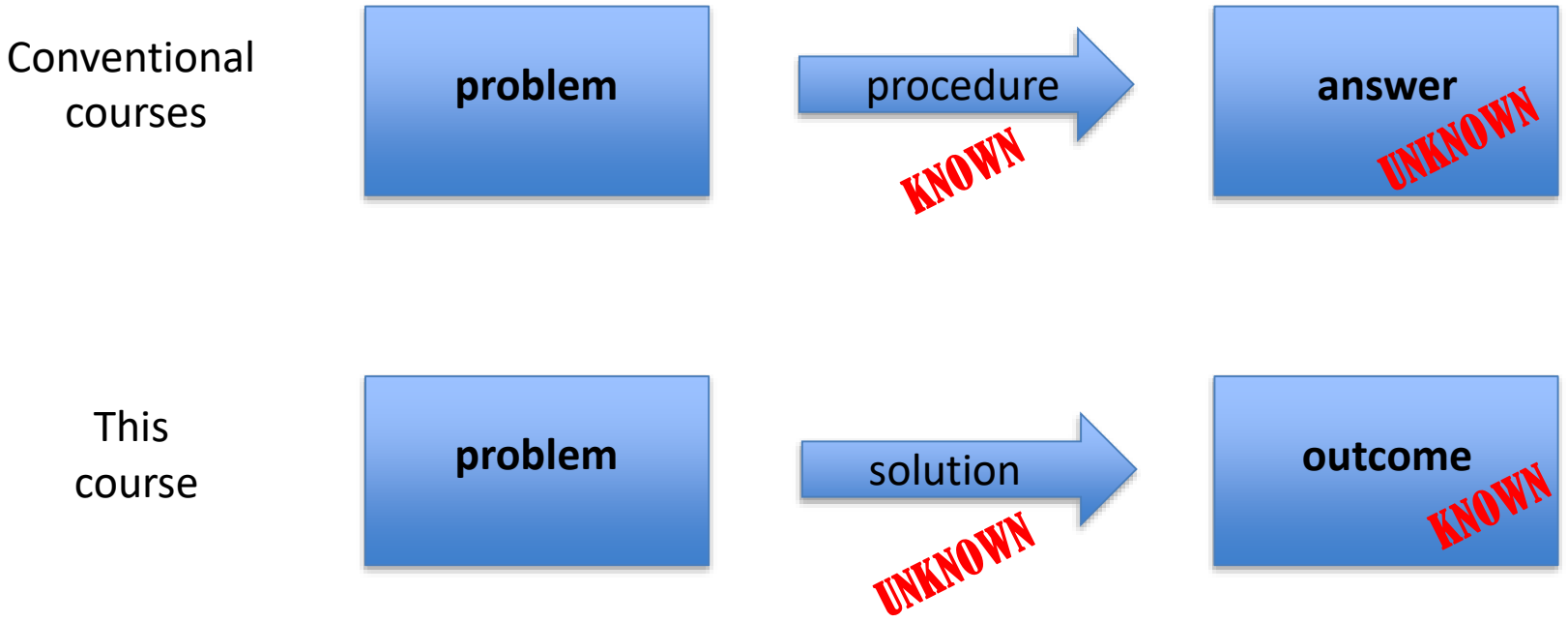
A word of caution

- 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.

UNDERSTANDING THIS COURSE

This course is not easy

- A different model for teaching



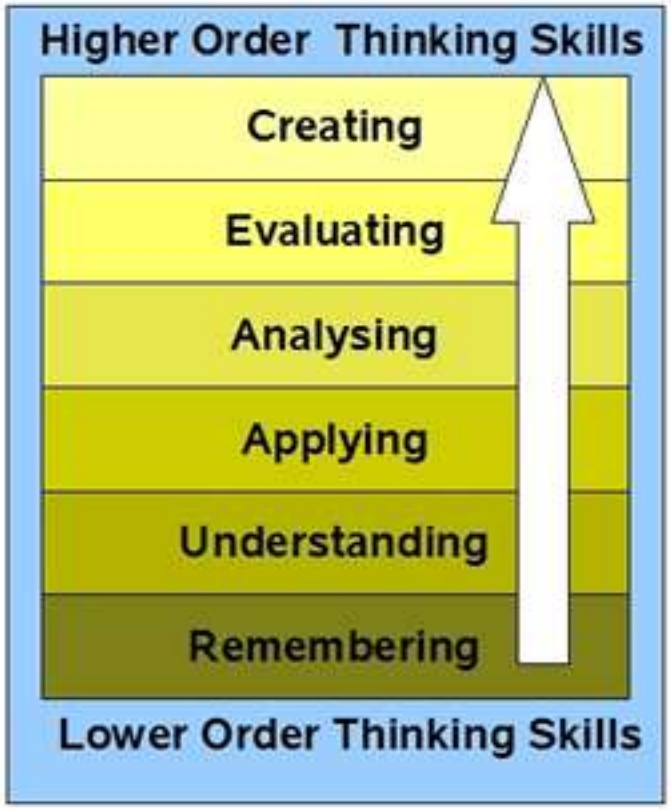
Slides and concepts based on work by Prof. Eric Mazur

Example

- You arrive with your car to a parking lot and there are no free spaces. You wait in a corner from which you can see (and control) 20 parking slots. How long do you have to wait before someone frees up a space?

- Solution requires
  - A model
  - Developing a model
  - Applying the model
  - Explaining the model and how it has been applied

The challenge in this course



Advanced Systems Laboratory

Bloom's taxonomie

The noise around ASL

- Do not let rumors drive you crazy
- If you hear, get, are given, or see advice from somebody: make sure it applies to your case and whoever is giving the advice knows what they are talking about
- Do not, under any circumstances, copy code, data, or results from anybody (we actually check but copying will not help you much)
- We evaluate mostly your understanding of the behavior of your system; it is very obvious when people present results about a system they have not built and/or do not understand
- Performance numbers are not random (you might have performance numbers but they could be wrong; at the very least you need to understand why they are wrong)

Thanks to the VIS

- … and the HoPo ASL taskforce for the feedback

- Read the latest issue of “Visionen”
  - “State of ASL”
  - “ASL Tips”