Scalable Systems and Development Processes

View PDF

Instructor(s):

Joseph Janos

Short Description of the Course:
A key engineering and entrepreneurial challenge is not only to quickly deploy the initial version of a great product but also, upon successful adaption by the market, to scale it up.

A product or product family can be scaled both “horizontally” by adding more feature sets to it and “vertically” attracting more users, on different platforms and in different markets, such as desktop and mobile, stand-alone and cloud, enterprise and consumer, local and international.

How and when to scale a product may be a business decision. The architecture of the code base and the engineering organization must be prepared for these requirements.

This course teaches both the software architectural and engineering organizational aspects of building large scale products. It emphasizes the dynamic, evolutionary nature of this process. Continuous innovation, scaling and adaptability are essential for successful companies. They should be prepared to build upon their existing products and engineering processes and organization.

The course teaches basic software architectural concepts, technologies and practices to architect products that can be quickly deployed but as the need arises can be smoothly, incrementally scaled.

Aim of the Course:
The aim of the course is to introduce to students best practices and technologies to build products that can evolve and scale over time. How to build products that grow from thousands of line of code to millions, developed and maintained from tens of software engineers to tens of thousands with a user base (supported load) from tens of thousands to tens of millions.

The course touches upon both engineering processes, such as source control, testing, bug tracking, monitoring and applicable technologies such as networking, load balancing, parallel computing, large scale data repositories.

Prerequisites:
The course requires thorough knowledge of big-O notion and analyzing algorithms using it. Knowledge of basic data structures (e.g. hashmap, lists, tuples) and operations on them (e.g. insert, find, sort) is required.

Methods of Instruction:
The course comprises a series of lectures, a team homework assignment and a talk presented by students from online resources.

Detailed Program and Class Schedule:

Part 1: Scalable Systems

Storage Technologies

- Non scalable storage. RDBMS, SQL.
- Distributed storage systems. NoSQL databases

- Bigtable, Dynamo, MangoDB

Networking

- OSI reference model
- DNS
- Load balancers

Distributed Computing

- Communication, synchronization, failure handling
- Leader election, consensus, mutual exclusion
- Paxos, Chubby
- Mapreduce, Pregel

Cloud

- Service Models: IaaS, PaaS, SaaS
- Cluster Management
- Bigdata
- Streaming data processing , lambda

Part 2: Scaling the engineering process

Design

- UML
- RESTful architecture

Coding

- Version Control
- Object Oriented Programming
- Agile, eXtreme Programming, Test Driven Development
- Refactoring

Testing

- White, black, grey box testing
- Unit testing, mocks and stubs
- Bug tracking systems

Integration and Release

- Acceptance test
- Configuration management
- Automated build and integration

Monitoring and Maintenance

- Service level agreements
- Signals, alerts, post mortem
- Distributed trace

Grading:
Instead of a final exam at each class there is a brief (5 minute) mini-quiz about the previous class. The final

grade is based on these mini-quizzes (30%), on participation (20%), on the prepared student talk (20%) and on the team project (30%).

Instructors' bio:

Joseph Janos received his degree in Mathematics from ELTE, Budapest, in 1973. He started his career in the Computer and Automation Institute of the Hungarian Academy of Sciences where he was Head of a CAD/CAM department. He left for the USA in 1981. He worked for the CS Department of SUNY at Stony Brook and for a number of large companies (Wang Laboratories, Lotus, IBM, Modicon.) In this period in various roles he architected and led engineering teams to build large scale back-end systems and UI intensive end-user products. (Lotus Notes and SmartCenter, a desktop electronic publishing system, a graphical front-end for PLC editing and monitoring.) After 1994 he worked exclusively on Internet-related technologies. Co-presence server with Ubique (sold to AOL), SurfLogic, his own startup, a client-side customizable crawler that he sold to Oracle in 1997, relevance matching engine (Lumapath) and a content delivery system (Radiance.) In 2004 he joined Google, where he was one of the first 25 engineers hired in New York. He retired in 2015. During his 11 years at Google he helped to grow the NY organization and led several engineering teams, building both consumer facing products (AdWords, Maps) and large scale distributed internal services (network monitoring and management, data processing pipelines, data mining.) As a senior level architect he learned and used most of Google’s vast technical infrastructure and technologies.