15-610: Engineering Distributed Systems (Spring 2013)
Syllabus and Course Organization

Instructors: Mahadev Satyanarayanan and Jan Harkes

Overview and Goals

The goal of this course is to give students the knowledge and skills required to create and evolve the complex, large-scale computer systems that society will increasingly depend on in the future. The course will teach the organizing principles of such systems, identifying a core set of versatile techniques that are applicable across many system layers. Students will acquire the knowledge base, intellectual tools, hands-on skills and modes of thought needed to build well-engineered computer systems that withstand the test of time, growth in scale, and stresses of live use. Strong design and implementation skills are expected of all students.

Although designated as a master's level course, it may be taken by well-prepared undergraduates with excellent design and implementation skills in low-level systems programing. The course assumes a high level of proficiency in all aspects of operating system design and implementation. Our goal is to impart both knowledge and skills. We want you to acquire a deep understanding of the engineering principles involved in designing and implementing mission-critical software. We also want you to be able to translate these principles into working systems. The course will consist of both lectures, where concepts will be discussed, and a lab, where hands-on experience will be gained. We hope that you find this an exciting, stimulating, and fun course in which you learn a lot of valuable knowledge and skills that serve you well throughout your career.

Course Topics

Caching for performance and availability

- origins of temporal and spatial locality, historical background
- versatility: hardware, local and distributed file systems, web, database
- design dimensions: level, granularity, persistence, timeliness (write-back/write-through, trickle reintegration), update safety
- challenges: maintaining coherence at large scale, rapid cache validation, failure resiliency of disconnected updates, transparency vs. translucency

Prefetching for performance and availability

- transforming high bandwidth into low latency (probabilistically)
- versatility: OS read-ahead, chunking or whole-file transfer, virtual memory replacement policy, hoarding, data staging
- dimensions: level of application, amount of advance notice, cache advantage, cost of mispredictions
- source of hints: explicit user advice, compiler directives, runtime observation
- challenges: interference with demand misses, mispredictions, resource hogging

Deduplication and Content-Addressable Storage for performance

- Cryptographic hashes: MD5, SHA-1, etc.
- versatility: rsync, LBFS, Lookaside caching, Pastiche, Centera, DHTs, Cedar
- challenges: collision resistance, computing overhead of hashes

Damage containment for reliability

- versatility: separate address spaces in Unix processes, AFS and Coda volumes, atomic transactions,

bad-block remapping on disks, packet-based file transfers, non-CS metaphors (watertight bulkheads, limited liability company, ripstop fabric)

- challenges: cost of boundary crossings (e.g.`read` vs. mapped I/O; cost of marshalling and unmarshalling arguments, threads vs. tasks), time and space cost of integrity checks, logging and state restoration, commit delay versus timeliness

Replication for availability

- origins and historical background
- challenges: update propagation, interpreting silence
- replica control: pessimistic and optimistic replica control, quorum-based strategies

Challenges of Size and Longevity

- **Scale reduction for performance and usability**
  - versatility: hierarchical naming, independent/autonomous name assignment, AFS and Coda volumes, rapid cache validation, wholesale resource allocation (eg. break() vs. malloc())
  - challenges: name space fragmentation, internal fragmentation
- **Hints for performance and availability**
  - self-validating data
  - simplified caching/propagation techniques
- **Reducing fragmentation for performance**
  - Why contiguity is important: substituting computation for data access
  - versatility: VM page size, file system block size, whole file caching, address translation, indirection, non-CS metaphors (primogeniture and entailing)

Designing for Human Foibles

- Limitations of individuals: attention span, reaction time, interactive performance and system latency, behavior under stress, cognitive and perceptual limits, trust limits
- Limitations of groups: spam, junk mail and postage, cloak of anonymity, tragedy of the commons

Hands-on Projects

A series of design and implementation projects are an integral part of the course. These are substantial projects that embody concepts taught in the lectures. The projects are done individually (i.e., not in groups).

Personal Hardware and OpenISR Parcels

We will loan you a laptop for the duration of the course. This will run a minimal Linux system at the bare metal level. On it will run the OpenISR system (http://isr.cmu.edu), which allows you to run, save and restore virtual machine images called *parcels* from an ISR server. All your project work will be done within parcels. You thus have 24x7 access to the lab hardware for this course, and don't need to compete with other students/courses for shared access. You have complete control over the machine including root access.

We will also loan you an Android smartphone for Project 4.

Please treat the laptops and smartphones as if they were your own, and return them in good condition at the end of the course so that they can be reused.

Course Resources

There is no textbook. Reference material will be handed out on occasion. Take good notes!

Course Web page

`http://www.cs.cmu.edu/~15-610`

Check here for handouts, updates, and so on.

Course AFS area

`/afs/andrew.cmu.edu/course/15/610`

Check here for class notes, software, etc. Remember, you need to be authenticated to the andrew.cmu.edu realm to access this material.

Course mailing list

`15-610@lists.andrew.cmu.edu`

This sends email to all students and instructors. This will be used for announcements by the instructors.

Faculty

Mahadev Satyanarayanan (Satya)
Contact information: GHC 9123, x8-3743, satya@cs.cmu.edu
Admin assistant: Angela Miller (GHC 9129, x8-6645, amiller@cs.cmu.edu)

Lab Instructor

Jan Harkes
Contact information: GHC 9127, x8-6658, jaharkes@cs.cmu.edu

Classes

Lecture/Lab:	Mondays, Wednesdays and Fridays
Time:	12:00 - 13:20
Place:	GHC 4211
First class:	Monday, January 14
Last class:	Friday, May 3
No class:	Monday January 21 (Martin Luther King Day) Friday March 8 (Mid-semester break), Monday-Friday March 11-15 (Spring Break), Friday April 19 (Spring Carnival)
Mid-term Exam:	Wednesday, March 6
Final Exam:	*to be announced*

Evaluation

A large part of your grade in the course will be based on the projects. There will also be a mid-term exam and a final exam, based on the material presented in the lectures and the required readings. A small part of your grade will be based your active engagement and participation in class (both lectures and lab). These components will be weighted as follows:

- Projects: 50%
- Mid-term: 20%
- Final: 25%
- Class participation: 5%