CS 7210 Distributed Computing
Spring 2014

- Instructor: Dr. Karsten Schwan (schwan@cc)
- Office: Klaus Building, Room 3338, Phone: 894-2589
- Office Hours: Monday 2:30-3, Wed after class
- TA: Ketan Bhardwaj (ketasnbj@gatech.edu)

Index

News & Notifications

There will be a class wiki. The initial versions of project proposals are due Jan. 18. Midterm and Final will be in-class exams, closed book, covering papers studied in class. A passing grade in the class requires students to have passing grades both in the midterm/final and in the course project components of the class, and class participation (and your lectures) counts heavily toward your midterm/final grades.

Course Description

Distributed computing systems have become pervasive. From clusters to internet-accessed data centers, to mobile machines, distributed systems are being used to support a wide variety of applications. This course will cover both fundamental concepts in distributed applications and systems enabling such applications. The following are the objectives of this course:

- In depth understanding of core concepts of distributed computing, bincluding study of both abstract concepts and practical techniques for building system support for distributed applications.
- Construction of distributed applications and supporting system components by doing project work.
- Understanding of current state of the art in one or more areas of distributed systems.

These goals are achieved as follows. First, class sessions cover a set of papers that discuss basic principles in distributed systems as well as systems and software infrastructures. Principles covered in the course include global states of distributed

computations, logical and physical clocks, and failure models. Distributed algorithms for consensus, replicated state management, and resource finding are also covered. Midterm and final exams feature questions on the papers covered in class. A second element of the class exposes students to representative distributed applications, systems, and infrastructures that support them. This includes wide area applications (e.g., peer to peer) and systems (e.g., DHTs, distributed stores), high performance applications or algorithms across high end machines and network links (e.g., cloud computing, using Georgia Tech's cluster machines and facilities), mobile applications running on pervasive platforms, and applications and infrastructures from specific domains (e.g., data streaming applications from the publish/subscribe domain). Programming projects evaluate students' knowledge for these practical class components. A third element of the class introduces students to rigorous research, since class projects are proposed and defended by students, and are documented in a form akin to research papers. These term papers are also presented in class. There is considerable flexibility in defining class projects and topics for term papers. Concerning projects, students can formulate their own, and depending on the nature of the project, they can work in groups of two. Finally, each student (or small group of students) is responsible for presenting one of the papers discussed in class. After successfully completing this course, students should be ready to explore research problems in distributed systems and on emerging applications deployed in such systems. This rigorous course requires strong class participation, and part of a student's grade is derived from class presentations.

Prerequisites

CS 4210 or equivalent. Graduate standing strongly suggested.

Grading

40% Examinations
40% Programming Project
20% Class Participation (includes class attendance and presentations)
A passing grade is required in each of the above components in order to pass the class.

Textbook and Other Readings

The class will primarily use journal and conference papers. ‘Background’ papers are not required reading; they are simply for your information. Some useful background appears in the Mullender and Singhal textbooks on distributed computing.

1. *Distributed Systems,* Sape Mullender, Addison-Wesley.
2. *Advanced Concepts in Operating Systems,* Mukesh Singhal.

Syllabus

The following papers on the listed topics will be discussed in class. Papers labeled as background are optional, for interested students. Each student will likely read additional

papers as part of the class project he/she is conducting. For papers without links, you should be able to download them from the ACM Digital Library or IEEE Xplore through the GATech Library.

Event Ordering, Global States and Time in Distributed Systems

Required background (see CS 6210): Lamport, L., " Time, Clocks, and the Ordering of Events in a Distributed System ", Communications of the ACM, 21, 7, pgs. 558-565, July 1978. (see CS 6210)

1. M. Raynal and M. Singhal. Logical Time: A Way to Capture Causality in Distributed Systems. IRISA Technical Report.
2. Schwarz, R. and Mattern, F., Detecting Causal Relationships in Distributed Systems: In Search of the Holy Grain, *Distributed Computing*, 1994.
3. Jeremy Elson, Lewis Girod, and Deborah Estrin, Fine-Grained Network Time Synchronization Using Reference Broadcasts, OSDI 2002 (presentation includes brief review of how network time protocols are implemented in the Internet - NTP).
4. Chandy, M. and Lamport, L., Distributed Snapshots: Determining Global States of Distributed Systems, ACM Trans. on Computer Systems, February 1985.

Abstractions for Supporting Distributed Applications I: Group Communication and DHTs

1. Ken Birman, Andre Schiper and Pat Stephenson, *Lightweight Causal and Atomic Multicast*, *ACM TOCS*, August 1991.
2. David Cheriton and Dale Skeen, *Understanding the Limitations of Causally and Totally Ordered Communication*, *ACM SOSP*, December 1993.
3. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H., Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications , TON.
4. Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, Ion Stoica. Wide-area cooperative storage with CFS, SOSP 2001. Also:Muthitacharoen et al., Ivy: A Read/Write Peer-to-Peer File System, OSDI 2002.

Failures and Distributed Systems

1. Intro: Survey of failures in distributed systems (Ch. 2, Mullender). Elnozahy et al., A Survey of Rollback-Recovery Protocols in Message-Passing Systems. Main paper: F. B. Schneider, Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial,, Computing Surveys, 1990.
2. M. Castro, B. Liskov, Practical Byzantine Fault Tolerance, OSDI, Feb. 1999.
3. Abd-El-Malek, Ganger, Fault-Scalable Byzantine Fault-Tolerant Services, SOSP 2005.

4. Dunagan et al, Fuse: Light-weight Distributed Failure Notification, OSDI 2004.
5. The **Google File System** (SOSP 03) and HDFS (Apache.org) (partial readings), and **Zookeeper**. (Apache.org)

Additional background reading:

Danny Dolev, Cynthia Dwork and Larry Stockmeyer, *On the Minimal Synchronism Needed for Distributed Consensus*, *JACM*, January 1987.

Tushar Deepak Chandra, Vassos Hadzilacos and Sam Toueg; The weakest failure detector for solving consensus, JACM 43, 4 (Jul. 1996), Pages 685 - 722.

M. J. Fischer, N. Lynch and M. S. Patterson, Impossibility of distributed consensus with one faulty process, JACM 32, 1985.

Other topics of interest: Reliable Multicast Protocols, Virtual Synchrony, Group Communication Systems, Real-time or other constraints. Publish/subscribe systems (e.g., Gryphon). Gossiping algorithms and their implementation: Trickle/Stanford;

Midterm Examination

Abstractions for Supporting Distributed Applications II: Replicated Objects and Consistency

1. H. Yu and A. Vahdat. The Costs and Limits of Availability of Replicated Services , SOSP 2001.
2. Ahamad, M. and Kordale, R. Scalable Consistency Protocols for Distributed Services, IEEE Transaction on Parallel and Distributed Systems. 1999.
   (Useful background: Mustaque Ahamad, Jim Burns, Phillip Hutto, Prince Kohli and Gil Neiger, Causal Memory, *Distributed Computing*, 1995.)
3. Bailis et.al., "The Potential Dangers of Causal Consistency and an Explicit Solution", SOCC 2012.
4. Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen, Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS, Proc. 23rd *ACM Symposium on Operating Systems Principles* (SOSP 11), October 2011.
5. Malkhi, D. and Reiter, M. Byzantine Quorum Systems , Journal of Distributed Computing, 1998.

Other interesting readings:

Gifford, D., Weighted Voting for Replicated Data, *ACM Symp. on Operating Systems Principles*, December 1979.

F. Torres, M. Ahamad and M. Raynal, Timed Consistency for Shared Distributed Objects , PODC 1999.

Terry, D. B. et. al., Session guarantees for weakly consistent replicated data, 1994 PDIS.

Danco Davcev and W.A. Burkhard. *Consistency and Recovery Control for Replicated Files*. In Proc. 10th ACM SOSP 1985.

Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer and Alan J. Demers, *Flexible update propagation for weakly consistent replication*. SOSP 1997 Pages 288-301.

Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Anderse, Stronger Semantics for Low-Latency Geo-Replicated Storage, Proc. 10th *Symposium on Networked Systems Design and Implementation* (NSDI 13), April 2013.

Naming, Resource Finding, and Distributed Infrastructures

1. Steen, M., Hauck, F., Homburg, P. and Tanenbaum, A. Locating objects in wide-area systems. IEEE Communications Magazine 1998. Also: David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris. *Resilient Overlay Networks.* Proc. 18th ACM SOSP, Banff, Canada, October 2001.
2. Landon P. Cox, Christopher D. Murray, and Brian D. Noble, Pastiche: Making Backup Cheap and Easy, OSDI 2002 (builds on Pastry).
3. Miguel Castro, Peter Druschel, Anne-Marie Kermarrec and Antony Rowstron, SCRIBE: A large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Select Areas in Communications, Vol.. 20, No. 8, Oct. 2002.
4. SDIMS: Praveen Yalagandula, Mike Dahlin, A Scalable Distributed Information Management System, ACM SIGCOMM 2004.

Other readings and P2P Systems:

P2P (Farsite, Samsara, Eigentrust and Strategyproof Mechanisms)

Mullender, S., Vitany, P., Distributed Match-Making, *Algorithmica*, No.3, 1988.

Other Topics:Infrastructures for P2P, Structured and Unstructured Overlays, Constructing Distributed Systems and Applications (e.g., Tapestry), FreeNet, Gnutella spec, P2P

Data-intensive Systems

1. Jeffrey Dean, Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, OSDI 2004. Plus BigTable (Fay Chang, et. al. Bigtable: A Distributed Storage System for Structured Data, OSDI 2006).
2. Spanner: Google's Globally-Distributed Database, OSDI 2012..
3. Marcos K. Aguilera, et. al, Sinfonia: A New Paradigm for Building Scalable Distributed Systems, SOSP 2007.
4. The Many Faces of Pub/Sub - ACM Surveys.

Mobility – will probably be elided for lack of time

1. Badrinath et. al., Designing Distributed Algorithms for Mobile Computing Networks. ICDCS.
2. M. Satyanarayanan, Fundamental Challenges in Mobile Computing, PODC 1995.
3. L. B. Mummert, M. R. Ebling and M. Satyanarayanan, Exploiting weak connectivity for mobile file access. SOSP 1995 Pages 143-155.

Other Topics: Algorithms adapted to mobile systems (e.g., radio issues, connectivity (e.g., reliable multi-hop routing), coordination in sensor networks, real-time communication and coordination, location management, end-to-end connectivity, structured (i.e., topologies) vs. unstructured (gossiping) communication, incentive-based approaches, ...).

Security in Distributed Systems - skipped due to overlap with Secure Systems Class

1. Lampson, B., Abadi, M. and Burrows, M, Authentication in Distributed Systems: Theory and Practice , ACM Transactions on Computer Systems, 1992.
2. M. Kaminsky, G. Saviddes, D. Mazieres and M. F. Kaashoek, Decentralized User Authentication in a Global File System , SOSP 2003.

Class Wiki

The Class Wiki provides a communication channel for all participants in the class.

Project Information

Guidelines

The purpose of a student-defined project is to (1) involve you in ongoing research projects, (2) leverage your unique background in some way, and/or (3) leverage other work in which you are involved. In general, any special project should be of a caliber that can generate results publishable in reviewed outlets like workshops or conferences (note that an actual publication typically requires some additional work beyond the time spent in this class). Your term paper is the prototype of a paper describing your project work.

Proposals

To propose your project for this class, you must submit the following materials:

- Brief project description:
  - purpose
  - solution approach you intend to use (experimentation and evaluation must be part of this approach)
  - expected outcomes/results
  - at least three different intermediate project steps, with delivery items and deadlines for each
  - final project deadline sometime during the week BEFORE the class' final exam (you propose a concrete deadline)

The first step must include both development (e.g., coding, experimentation) and background work, such as producing a bibliography of relevant papers and having read them, having designed suitable algorithms/approaches, and having learned or looked at suitable tools to be used for your project, including target platforms.

The second step, typically after the class midterm, should involve having produced much of the software necessary and having debugged it.

The third step should include not only software testing but also evaluation, on the platforms you have chosen. Such evaluations may include theoretical results if you choose to develop or experiment with a novel algorithm, for instance, but it should also include experimental results.

The final deliverable not only includes the actual software but also a report – the term paper -- which is outlined next.

Final Report

The on-line final report regarding your project should have the following parts:

- A statement of your approach to the project and the technique used to solve it -- two typewritten pages minimum, 8 pages maximum, including a list of references to related work and a discussion of related work.
- If applicable, a running program with sufficient documentation so that someone can understand your program without re-running it. Such documentation should consist of comments within the program text and of separate explanations (e.g., readme file). You should be prepared to hand in your program electronically, if requested. Most likely, you will simply schedule a demo with the instructor.
- A conclusion, stating the main results of your work. This conclusion might contain a performance evaluation of your program or a list of next steps concerning it (what might be interesting to do next).
- How your work should be extended, what should be done to make it more useful. Maximum 4 pages, minimum 1 page.
- A one page evaluation of what you did: its usefulness in the context of other work and in the context of general research (namely, why did you do this and why was or wasn't it worthwhile doing?). This is where there's a potential linkage to your term paper, as well, but please just state the linkage, don't repeat the term paper.
- A one page discussion relating the work you did to the topics we studied in class. Comment on what papers in class relate to what you did or to extensions of what you did, if applicable.

Facilities Available for CS7210 Projects

1. 'Hack' cluster (virtual machines running on server systems):
    - Ability to get superuser privileges to do kernel-level development.
    - Enforcement of real-time constraints possible through ability to have privileges akin to super-user
    - Ability to run on dedicated single (2-way shared memory multiprocessor) or multiple machines, networked via switched 100MB or Gigabit Ethernet.
    - Ability to use experimental kernel-level real-time facilities.
    - Ability to experiment with virtualization technologies, including Xen.

The class TA will be able to provide you with more information.

3. Remote access to NetLab cluster machine (requires separate process for account creation and approval).
4. Use of PlanetLab – also requires separate account approval process.
5. . Cloud computing facilities – we have a large VMware vCloud setup running; we have an OpenStack cloud.

Contact Chad Honeycutt (chadh@cc) for more information and machine access.
6. 'Enterprise' or HPC computing clusters: potential access to several cluster computers used by systems researchers and located at Georgia Tech.

Sample Exam

2007 MidExam