CS 441: Engineering Distributed Objects for Cloud Computing
Fall 2021

Course Instructor:
Name: Dr. Mark Grechanik
Email: drmark@uic.edu
Office Hours: TTH, 3:30PM-4:45PM or by appointment

Class Meets:
TTh, 2:00 pm-3:15 pm at Lecture Center Building F F001

TA: Mr. Vineet Rajesh Patel. He will announce his office hours via the Blackboard.

1. Course Web Address:

All course materials will be made available to you via Piazza. Grading and some materials will be available via the university Blackboard (BB). It is your responsibility to set up your account with the BB and Piazza and use it to read course announcements, view your grades at BB and participate in discussion groups at Piazza. If you have problems using the BB or Piazza, then please use the corresponding help desk to resolve them.

2. Prerequisites:

- Firm understanding of the software development steps. Knowledge of source control systems (i.e., Git) and integrated development environments is required. We will use IntelliJ for homeworks and projects in this course, so each student must obtain its academic free version and get familiar with it. We will use Sbt to build and run projects. All submissions will be done via Bitbucket.
- Students are expected to have taken at least one of CS 341 or CS342 and CS 361, and preferably CS 385 OS with a grade of C or better or their equivalents. Good knowledge is required of the course Language and Automata (CS 301) and the Data Structures course, CS 251.
- Familiarity with object-oriented programming, specifically C++ and/or Java and and/or Scala and/or and/or C# Go and/or Clojure and/or Elixir and/or Haskell. Knowledge of functional programming is a BIG plus.
- You should have access to a computer with at least 16Gb of RAM with at least a double-core 2GHz CPU and a 200GB+ free space on its hard drive to which you should have the administrative access. In general, as a CS student taking an advanced systems course like this one you need a more powerful laptop, which you could rent or purchase. NOT having a sufficiently powerful laptop or a workstation or a server is akin to NOT having a good quality violin for a professional practicing violinist! With starting salaries in our area in the range of $90,000 per year , purchasing a $3K+ workstation is not a luxury, it is a necessary investment in yourselves.

- It is important to come to lectures rested and prepared to participate in discussions. Lecture attendance is optional, however many students attend because they can interrupt the lecture to clarify the material directly from me, the lecturer. Therefore, if the student does not follow the lecture, there is little point in her/his sitting in the classroom. Some students suffer from micro-sleep (MS) defined as "a temporary episode of sleep or drowsiness which may last for a fraction of a second or up to 30 seconds where an individual fails to respond to some arbitrary sensory input and becomes unconscious. MSes occur when an individual loses awareness and subsequently gains awareness after a brief lapse in consciousness, or when there are sudden shifts between states of wakefulness and sleep. In behavioral terms, MSs manifest as droopy eyes, slow eyelid-closure, and head nodding.[1]" MS will interfere with your ability to remember to rest well before you come to lectures.

  If you have not met these prerequisites, email to me or talk to me about it.

3. **Required Textbook(s):**

   Mark Grechanik. Theory and Practice of Cloud Computing. The PDF version of this book can be purchased from payhip. I will also give students reading assignments that include research papers, articles, and handouts. Reading assignments are mandatory! You will be able to integrate your class notes with copies of slides that I use in class and with papers and web sites relevant to the material discussed in class, all of which I will post on Piazza under Resources. There is no DRM that would otherwise make this book more expensive, so once purchased, students will email the order number or, alternatively, they can use optional books. I will assign exercises and exam questions based on the books you use to study.

4. **Optional Textbooks:**

   These textbooks will not be used in the course for any mandatory reading assignments, however, they may help students understand course topics better.

   Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services (Texts in Computer Science) by Kenneth Birman.

   A.D. Kshemkalyani, M. Singhal, Distributed Computing: Principles, Algorithms, and Systems, ISBN: 9780521189842, paperback edition, Cambridge University Press, March 2011. 756 pages.

   Vijay K. Garg, Elements of Distributed Computing, Wiley-Interscience, 2002.

[1] https://en.wikipedia.org/wiki/Microsleep

5. **Course Content:**

Large complex software applications require sophisticated hardware and software infrastructures in order to deliver high quality services to their users. Historically, many organizations have chosen to own, manage, and operate their infrastructures, bearing the responsibility for guaranteeing that they are sufficient and available. As a result, such organizations must periodically evaluate their infrastructures and forecast their resource needs (i.e., CPUs, memory, storage). Armed with these forecasts they then purchase and maintain the resources required to meet those needs. Since it is difficult, if not impossible, to exactly predict resource needs, especially over a long period of time, these organizations often end up grossly over-provisioning or grossly under-provisioning their infrastructures. And therefore, they have found it difficult to cost-effectively provide for their long-term computing needs.

Cloud computing offers an alternative to the in-house infrastructure. With cloud computing, organizations deploy and run their software applications on a sophisticated infrastructure that is owned and managed by external cloud service providers (e.g., Google Cloud, Amazon AWS). The goal of this course is to provide a broad but solid overview of engineering distributed object programming for cloud computing, which is a discipline of designing, implementing, and maintaining software systems that are composed from programming objects that do not share the same state. Since software that uses distributed objects has become ubiquitous in today's world, it affects all aspects of our lives. However, building this software requires significant resources and knowledge, and often ends as a failure.

In this course we review different areas of distributed object with the concentration on practical aspects of using cloud computing to support various properties of distributed objects. Lectures will be based on the material from the textbook, accepted standards, and on research papers written by accomplished software engineers and computer scientists. Students will be introduced to various topics of distributed object programming, and different domains to which methods and tools learned with each topic are applicable.

6. **Course Goals:**

The goal of this course is to provide a broad but solid overview of engineering distributed object-oriented applications for cloud platforms, specifically with in-depth coverage of designing, implementing, and maintaining software systems that are composed from programming objects that do not share the same state in cloud environments. Since software that uses distributed objects has become ubiquitous in today's world, it affects all aspects of our lives. However, building this software for cloud requires significant resources and knowledge, and often ends as a failure.

In this course we review different areas of engineering distributed object with the concentration on practical aspects of using cloud platforms to support various properties of distributed objects. Lectures will be based on the material from accepted standards and on research papers written by accomplished software engineers and computer scientists. Students will be introduced to various topics of engineering distributed objects for cloud computing, and different domains to which methods and tools learned with each topic are applicable.

The course will consist of readings, quizzes, two exams, and a final project. The goals of the course are to:

- Understand the theory of distributed objects;
- Learn principles behind cloud computing using distributed objects;
- Cover various approaches and standards for creating and using distributed objects for cloud computing, and to
- Enable students to design systems of distributed objects using various tools and methods with popular cloud computing platforms.

7. Course Topics:

Please note: reading materials for the next week classes will be provided on Friday of the previous week. It is expected that you read these materials before coming to lectures, so that you can actively participate in the class.

A. Overview of Key Drivers of Cloud Computing.
B. From Diffusing Computation to Cloud Computing.
C. Cloud Model and Operational Environments.
D. Remote Procedure Calls.
E. Cloud Computing with the Map/Reduce Model. Google File System and Apache Hadoop.
F. RPC Galore: Java RMI, XML-, GWT-, and JSON-RPC, Facebook Thrift, Google RPC, Twitter Finagle, Facebook Wangle.
G. Cloud Virtualization. Virtual Machines. Hypervisors. Lock Holder Preemption. Virtual Networks.
H. Software Appliances for Cloud Computing.
I. Cloud Infrastructure.
J. Scheduling and Balancing the Load for Distributed Objects in the Cloud. Google Borg and Omega. Microsoft Octopus.
K. Spark Data Processing.
L. Web Services.
M. Microservices and Containers. Docker, Kubernetus.
N. The CAP Theorem.
O. Soft State Replication. Eventual Consistency.
P. Facebook's Algorithm for Computing Eventual Consistency.
Q. Digital Currency. Blockchain. Bitcoin.
R. Data Stores in the Cloud. NoSQL.
S. Overlay Networks. Chord.
T. Yahoo PNUTS.
U. Amazon Dynamo.

V. Cloud-based DevOps.
W. Facebook HipHop VM.
X. Quantum Computing in the Cloud.
Y. Cloud Economics.

8. **Grading and Bonuses:**

- Piazza and class participation: 5%
- In-class quizzes: 10%
- Homeworks: 25%
- Midterm: 20%
- Final: 20%
- Project: 20%

I grade on a curve and in addition to grade points earned for the exams and assignments, I give bonuses to students for their contributions to problem solving in the classroom and on piazza. The bonuses are not counted towards the final grade at first, then the final grade sans bonuses is used to create a curve, and then add the bonus points to move students across the grade categories. In general, the distribution of bonuses in class is close to exponential with a long decaying tail where a small number of students get high bonuses and many students get very small or zero bonuses. It is highly likely that students with high bonuses study hard and their grades are also high.

9. **General:**

- You are responsible for knowing the academic calendar and adhering to its deadlines, including but not limited course add/drop dates and payment due dates. All differences between information in this syllabus and the official UIC academic calendar should be resolved in favor of the calendar. Please notify Prof. Grechanik about the differences if you discover them.
- Some in-class lectures will start with a 5-10 min quiz without prior announcements.
- Bonus points will be given in class for active participation in lectures.
- Lecture attendance is not required, you are responsible for learning.
- Eating, sleeping, browsing the Internet in the classroom are permitted as long as these activities do not distract other students and me and do not violate UIC policies.
- Our class does not meet during the Thanksgiving break and during the Spring break, when applicable.
- Please check the academic calendar for the last day to complete late registration; last day to add a course(s) or make section changes; last day to drop individual courses via Student Self-Service without receiving W (Withdrawn) grade on academic record. Make sure that you know the last day to submit Withdraw from Term request via Student Self-Service and receive 100% cancellation of tuition and fees.

- Midterm will be held during the regular lecture hours on Thursday, October 15, 2020 via the lockdown browser.
- The day of the final examination will be announced later – the university will schedule it.[2]
- You must attend the midterm and final exams and complete your project to satisfy the baseline requirements set by me to be considered for a grade of C or better.
- No makeup exams and quizzes will be permitted unless authorized by the CS (under)graduate advisor or the department head.
- All homeworks and the course projects should be submitted via Bitbucket. Submissions by email and BB will not be accepted.
- The course project submission date is Wed, December 8, 2021 at 10:00PM via Github or Bitbucket. Failure to submit your project by the deadline shall result in losing your project grade.
- You can discuss your grades with me only if you failed to resolve the issue with the TA and if and only if you ask questions about specific points you lost in your quizzes, exams, and the project.

Examples of homeworks – real homeworks can deviate from these examples:

HW1: create a cloud simulator using CloudSim given a specification.

HW2: implement HW1 using gRPC in GO. The client should create a document in the Google cloud and populate it with information about its requests and responses.

HW3: implement the server in HW1 as a REST service using Scala and Http4s, a Scala interface for HTTP services. Test it with the Postman application.

HW4: Extract stock quotes from a web service, send the stream of these quotes to AWS Kinesis, process the stream to compute statistical indicators of selected quotes.

HW5: Create a distributed software application for automatically analyzing the content of research papers in computer science using a map/reduce model. You will create and run your software application using Apache Hadoop (http://hadoop.apache.org/), a framework for distributed processing of large data sets across multiple computers (or even on a single node) using the map/reduce model. To run the VM, you can install vmWare or VirtualBox. As UIC students, you have access to free vmWare licenses, go to http://go.uic.edu/csvmware to obtain your license. You can complete this homework using Java and you will use SBT for building the project and running automated tests.

Example of an individual course project: using your google credentials, create a client application that connects to the google store and downloads APK files and metadata. As the files are downloaded, create Akka Actor-based services for analyzing these APKs and extracting and summarizing information from them. Feed this information into a service that uses the Weka framework to learn patterns from attributes associated with the APKs.

[2] The dates for the midterm and the final are subject to change. Stay tuned for announcements.

How to Study and Excel in CS441 or a Similar Systems/Software Course:

Courses in systems/software are much more complex than a garden-variety textbook theory course because in addition to learning the terminology, hypotheses and proving theorems students need to combine and to apply the learned concepts to create solutions as working software. In a way, most nontrivial problems in systems/software are ill-posed (https://en.wikipedia.org/wiki/Well-posed_problem), since exact solutions may not exist or many approximate solutions exist and choosing one requires in-depth analyses of the problem; moreover, slight changes in the configurations of the system may render a chosen solution incorrect or inapplicable.

The starting point is to read about a given topic, to study the terms and definitions used it the statement of a problem, to understand the key driving forces that create the problem. Review the solution to this problem and trace mentally how the solution uses the terms and definitions used to describe the problem. Your goal is to capture the basic ideas for the presented topics and it is ok to miss some details.

If you cannot understand some parts of the problem description or the solution, trace it back to the definitions and the terminology to pinpoint the location where your comprehension breaks down. Doing so will help you formulate intelligent questions, e.g., "I understand how messages are formed into arrays, but since there are no clearly defined guards between messages, how will the receiver determine where these messages are separated?" An opposite, non-intelligent or haphazard question may sound like this: "I read this chapter twice and I have no idea how to write a program that sends and receives protobuf messages. What do I do?"

Given the breadth of the topics and prerequisites it is important that you dedicate some time to self-study, specifically to cover technical concepts and techniques that are important for completing your assignments. For example, there is plenty of videos and other documentation that explain how IntelliJ IDE works from grounds up. Depending on your level of preparation and how you match prerequisites for this course you will need to use these learning materials to get up to speed on some topics. The main material is presented in my book and class lectures, however, you should be highly proactive to determine what other material you need to learn quickly to make this course effective for you. It is unreasonable to dedicate precious lecture time to learning, for instance, how SSH works and how to configure it and use its commands. Self-learning discipline will help you navigate the world of systems and software effectively.

After you understand the basic elements of the problem and its solution(s), it is time to apply it to gain in-depth comprehension, something that allow you to explain to yourself and the others what will happen with a solution if you change some parameters. In systems/software, you start with small practice problems that you can work out on the paper and then you move to implement it in software projects, deploy them using a debugger and play around with it to understand the cause and effect of your changes. In short, I advise you to use the famous Feynman technique popularized here to break down the solution into small steps and determine what exactly they do to solve the problem.

Once you absorb the terminology and the usage of all definitions to the point when your brain does not require you to reference the textbook when you recall a term/definition, you need to internalize the solution so that you know why it works on the intuitive level. There are many videos, websites and books dedicated to giving intuition why some solutions/approaches/techniques/algorithms work – consider as an example the following blog entry on Genetic Algorithm (GA): A Simple and Intuitive Guide.

At this point, you can apply your newly acquired knowledge to solve corresponding problems in your course homeworks and the project, which are described loosely enough so that you have flexibility to play around with the algorithm/technique that you learned. You can experiment, change parameters, combine your learned solutions in new ways and report your results and experiences on Piazza for which you will earn additional points. You will experience the level of mastery that brings your immense joy of creation and being in control of the systems that you build.

Finally, you can join some open-source projects and start contributing your solutions by fixing reported bugs and offering more effective solutions to already solved problems, where you may improve the efficiency of the software or reduce the amount of written code by refactoring it. You may want to create a new framework or a library and earn your followers. More importantly, you will keep gaining deeper understanding of the material in this course that you could ever imagine!

Fairness:

It is a fundamental right of each student who takes this course (and other courses) to be treated fairly. Especially, in a large class we have students with different backgrounds, family/work situations and various personal restrictions: there are students who are single mothers and fathers with full-time jobs who need to care for their children, military and other civil servants who risk their lives as part of their jobs, and students with disabilities who need special accommodations, to name but a few. My job is to ensure that this course proceeds smoothly and it is your responsibility to do your assignments and attend lectures and perform all course duties by organizing all your activities to ensure that all submissions are done on or before the deadlines and you take all exams in a pre-arranged manner according to the schedule.

Fairness is violated when you ask me to allow you to submit your assignments late or make up for an exam for you, specifically, whatever reason. It means that if I allow you to do it, I would treat all other students unfairly, because they managed to organize their lives to postpone other important tasks, e.g., to spend time with their children, at some nontrivial expense to them (e.g., hire a babysitter) to complete the assignment on time. Unfair treatment of students breeds resentment and prevents us from having a healthy learning environment.

The only reason to postpone the submission deadline can be done by offering solidly documented and verifiable emergency that occurred to you, e.g., the verifiable documentation of bodily injury from an emergency room with the waved HIPAA rights, so that our department personnel can obtain an independent verification of this emergency. Once a student gave me a jury service notification where the date of his appearance fell on the midterm. I wrote a letter for him explaining that he had a midterm on the date, and his jury appearance was rescheduled. Please take the issue of fairness close to your heart when you think about asking me to postpone a submission deadline.

Emergencies and Deadlines

As the deadlines for homework/course project and the exams get closer, I get emails from some students about their visits to doctors/emergency rooms with requests to postpone the deadlines for them. Here are some general rules. To accommodate you, I need to go through a process of verifying this situation. This process is the same for all students and it involves a few steps. If you provide a doctor's note to me, be aware that I can contact the doctor or hospital to verify if you were seen on the date(s) in question. If your doctor is unable to verify your visit without a signed HIPAA authorization form, you may need to complete a HIPAA authorization for release of health information form in order for me to verify that you were seen by the doctor on the date in question.

In general, I do not need to see a document that describes a medical condition that brought you to a doctor. The attending doctor, however, should provide official evidence that a student’s condition prevented the student from taking a given exam, including potentially via the UIC dean of students. Once this verification is done, we can discuss the accommodations based on its outcome.

Know Your Computing Environment and Learn to Debug It!

Many students in this class will experience difficulties setting up the development and deployment environments for your homeworks and the course project. I would like to offer a few tips to alleviate this problem, and I start with an analogy to compare each of us with a violinist or pianist or some other musician. Or we can use an analogy with a surgeon.

Imagine a violinist who complains that s/he spent a lot of time fine-tuning her/his instrument because s/he didn't know how to tweak different knobs on her instrument. Or a surgeon who complains that instead of cutting flesh and replacing a bad heart valve s/he had to figure out how to set up and use tavi-midcab retraction system on a patient's heart. You can reply that musicians must learn to fine-tune instruments before playing complex pieces and surgeons should not be allowed to cut flesh until s/he learns how all instruments work and how to use them. Using this analogy, it will leave us with the following question: how much cloud computing material we could cover if lectures are spent on showing how all those tools work in practice.

As computer science students, you should know your hardware (your laptop or your workstation) and all the software that runs on it. You should also learn about well-known tools that automate various tasks. Just like a violinist knows her violin and constantly experiments with its sounds and knobs (it is a right term?), you should know your tools and constantly learn about new ones.

Given the number of tutorials and discussion board and youtube videos and safaribooksonline database, it is not justifiable for professors to use the classroom time to go over the tools and explain basic and well documented operations on the Internet, e.g., how to set a breakpoint or how to debug an external Java process in IntelliJ. Understanding how source control systems work is important. In one situation, a student pushed the code into the master repo mistakenly and the other student pulled this code into his local repo overwriting his local settings. Both situations resulted from the lack of attention, lack of consideration, mental laziness, and the desire to cut corners instead of figuring out how Git works in detail. As a result, one student lost a few days of work, so he paid with his time for the light attitude towards the complex systems.

It is a part of the learning experience where you develop and refine your abductive reasoning skills. You observe symptoms, recall the set of actions that led to the appearance of these symptoms and then you create hypothesis of what action can cause these symptoms. Then you think of experiments where you modify the system and see how the symptoms change. Slowly, you will converge to a solution.

In case of the example above, this approach would work as the following. The student will delete the non-working project and create a new virgin project and
add libraries and write some basic code. Once he sees it is working, he will pull the repo and add the working code to the repo and push it. Then he will pull the code from the repo again into a new directory and experience the problem. Now, there are two scenarios: the virgin project works, and the one merged with the repo doesn't. The same idea applies to the other problems you have experienced.

Of course, to do that you need to apply your brain to design a systematic approach to your engineering complex applications. If your attention span is short and you are looking for shortcuts, you will end up very frustrated in this course.

Selecting Teammates for the Course Project

Your course includes a course project. Regretfully, every time I teach this course I receive complaints shortly before project submission deadline that some team members didn't do anything and the working team members want justice by excluding the freeloaders from the team. This is the reason that I am sending this message - to warn you about possible dangers of forming teams and how to ensure that your course project work is productive and enjoyable.

Ask your prospective teammates to give you read access to their homework repos and in response you give them access to yours. This way you can see their previous deliverables and gain confidence in their skills.

Define project milestones early on and determine who is responsible for what. Do it in writing via email or set up a project in basecamp (you can get an academic license).

Document progress. If your teammates do not respond to your email inquiries about their progress on milestones, notify me after two or three attempts to contact your teammates by email. Include your email exchange and CC to your nonresponsive teammates. With this evidence, I can intervene and break up the team if your teammates did not deliver as agreed. To make it effective, you need to set up milestones early in the project lifecycle and keep the documentation to prove the pattern of nondelivery.

Interviews:

On more than a few occasions, I received emails from students or verbal requests to delay quiz/homework submission/exam/... because these students had scheduled interviews or they planned to attend career fairs (with possible interviews there). In some cases, these requests were given to me in a tone that suggested that since a company like Google selected a student for an interview, I should have acknowledged the greatness of the process and humbly step aside with my pestering course assignments. Responding to every request like this takes time and effort, so I want to do it in this syllabus.

You are invited to interviews, most likely because you are studying at UIC CS and taking courses and doing assignments, so your interviewers think that you stand out from the crowd because of your education. They will respect that you take your courses seriously and they will accommodate you gladly, since it shows that you value your commitment to your current projects more than your hunt for future projects. When I asked my friend, a manager at Google about students postponing interview because of the coursework, this is what she replied to me: "In my own experience the recruiter would ask for the student's availability before schedule (sic) an interview. Even after that the date is not carved in stone. Unless the student is facing deportation by USCIS without a job offer by certain date, I don't see a pending interview as a reason to ask for submission deadline extension."

Office Hours Policy:

Some of students come to office hours with a laptop in their hands asking your TA or your professor to figure out why the code that you wrote does not work. I emphasize that **neither the TA nor your professor will troubleshoot your failing code during the office hours**. There are three reasons for it.

1. In some cases it is easy to see the fault in your source code or your project configuration whereas in many other cases the troubleshooting requires checking a number of external variables (e.g., the classpath), determining whether you specified proper dependencies, analyzing your build environments and versions of external dependencies used in your code and last but not least making sure that you correctly installed required software platforms - all that takes a significant

amount of time that requires other students to wait and doing so is not properly suited for office hours.
2. Helping one student resolve problems with the code would require us to do the same for all students to be fair, which is not scalable for the class of 50+ students.
3. If we fix problems for students, they never learn. Our goal is to help you, not to do the work for you.

A bigger picture is that you will not be able to approach your coworkers and ask them to solve your problems unless you can show to them that they are a part of the problem (i.e., your program fails because of the computation that is performed by a program written by your coworker). You need to explain clearly what you do and why you think your program fails.

To do that you need to make sure that you follow these basic steps.

- Make sure that you create a virgin sandbox environment where you can test a specific function. For example, if you want to test file I/O using the JDK, make sure that you can create and compile a basic Java program with one class with the method main and run it to print out a basic message. Then you add one function that opens a file. Make sure that the file is at the destination and it has proper security and other related attributes. If your program fails, check the exception/error message and determine if your computing environment is at fault.

- If you use some external software dependency, make sure that you are using a proper version and it is resolved by your build system. Find a basic example in the documentation for this software and go to the previous step and repeat this example. Make sure that you start from scratch when going though the setup.

- If everything fails, you can take your small example and post it on piazza or put it into a public repo in bitbucket or github and share a link to it on piazza.

Cooperation policy:

I encourage you to discuss the problem sets and programming assignments with your colleagues. I welcome discussions of possible interpretations of questions, solution approaches, and relevant technical details. You are also welcome to use existing public libraries in your programming assignments. Such activities qualify under approved collaboration practices and you are welcome to take advantage of them.

Note that cooperation is not the same thing as cheating. It is OK to ask someone about the concepts needed to do homework or project assignments. However, ***copying*** other people's code or solution sets is strictly prohibited. The quizzes, project assignments, and exams must be the work of students turning them in. Students who violate University rules on scholastic dishonesty are subject to disciplinary penalties, including the possibility of failure in the course and/or dismissal from the University. Because such dishonesty harms the individual, all students, and the integrity of the University, policies on scholastic dishonesty will be strictly enforced. Acts that exceed the bounds defined by the approved collaboration practices will be considered cheating.