SYLLABUS

****

YEAR COURSE OFFERED: 2022
SEMESTER COURSE OFFERED: Fall
DEPARTMENT: Computer Science
COURSE NUMBER: 4397 (6397 for grad section)
NAME OF COURSE: Systems Programming: Big Data
JOINT SECTION: Yes (6397)
FORMAT: Hybrid (75% face to face, 25% online)
NAME OF INSTRUCTOR: Carlos Ordonez

****

The information contained in this class syllabus is subject to change without notice. Students are expected to be aware of any additional course policies presented by the instructor during the course.

****

Learning Objectives

Students taking this course will learn theory and programming mechanisms to store, update, query and analyze data at large scale: without memory limitations and in parallel. Students will advance their knowledge of modern relational database systems, internal storage and runtime in data science languages and data-oriented parallel processing. Students will gain an understanding how algorithms, data structures and mechanisms work in C (Linux, legacy code) and C++ (new generation DBMSs, stream systems).

Major Assignments/Exams

The two major items and their weight in the final grade are:

- 80%: Programming project using the C and C++ programming languages, developed in 2 phases. This project will require getting familiarized with the internal subsystems of a modern database system, parsing and evaluating queries (SQL) as well as understanding the code evaluation runtime & storage of a data science language (e.g. Python, R). This project will be developed in teams of 2 students.
- 20%: Midterm exam: 10 questions, open-everything, in-class, written answer, grade [0-100]. The exam will take place around the 10^{th} week of class. The weight of the exam may be decreased, depending on the quality of programming project.

Differences between undergrad and grad sections.

Experience with 3380/6340 (Database Systems) and 4315/6345 (Programming Languages) shows undergrad students tend to be better C++ developers than grad students. On the other hand, some grad students tend to be better at theory/math concepts

and they are generally more agile with Python and its libraries. Since some questions in the midterm exam require mathematical and programming maturity the weight of the exam can be reduced to 10% for undergrad students, if their projects satisfy all requirements. On the other hand, for grad students the weight of the exam can be increased to 30% if the exam score is good (i.e. it has a grade 60 or higher), but their C++ projects are incomplete (it fails, incomplete). Such adjustments would happen after the midterm exam is graded.

Reading

Since processing big data is a dynamic field, references and textbooks change continuously. The references below are recommended, but not required.

Readings in Database Systems, 5th Edition. Peter Bailis, Joseph M. Hellerstein, Michael Stonebraker, editors. This is an amazing collection of papers collected by well-known researchers from MIT and University of California at Berkeley. The PDF is freely available on the web http://www.redbook.io/.

Students will get familiarized with popular websites of big data and large-scale data processing, some from academia, some from industry. The course will encourage reading about 20 papers from mainstream CS conferences and journals, focusing on large-scale data processing like IEEE Big Data, ACM SIGMOD, Dawak, DOLAP, Distributed and Parallel Databases, Intelligent Information Systems, IEEE TKDE, ACM TKDD. About 25% of these papers are coauthored by the instructor and his students.

All papers can be accessed from DBLP, with direct links to the PDF inside UH: https://dblp.uni-trier.de/.

Open-access versions PDFs of the instructor's papers are available at:
http://www2.cs.uh.edu/~ordonez/co_research_proceedings.html
http://www2.cs.uh.edu/~ordonez/co_research_journal.html

List of topics

The following topics take about 12 weeks of lectures. The 2 additional weeks are a buffer to make adjustments and the midterm exam.

1. Main theoretical differences between relational database systems and non-relational data systems to store and retrieve data.
2. Advanced C and C++ programming including byte-level storage of simple data types, pointer manipulation, binary I/O, thread synchronization, coding standards, compiling large projects, debugging runtime errors
3. File system mechanisms to store large files. Common file formats to exchange big data (CSV, JSON), displacing XML

4. Advanced SQL: partitioning tables, recursive queries, UDFs, OLAP functions.
5. Block-based I/O (records) algorithms and data layout. External search and sort algorithms (low memory footprint).
6. ACID transaction data processing requirements and how different data systems relax them. Locking vs timestamping.
7. CAP theorem showing the compromise between availability and fault tolerance and why no distributed system can be perfect.
8. Hashing: functions, distributed data partitioning, managing collisions on I/O blocks.
9. The need to develop alternative data-oriented algorithms to manage main memory, complementing OS algorithms
10. External data structures to store relational tables, matrices, text and web data
11. Contrasting parallel processing in multi-core and multi-node architectures
12. Distributed file systems versus partitioned tables (e.g. HDFS vs parallel SQL engine).

Pre-Requisites and Overlap with other courses

Pre-requisites: A student should have taken all 2000 courses or equivalent for grad students. It is desirable students have taken most fundamental CS 3000 courses including: Algorithms, Database systems and Operating Systems. Programming experience with C and C++ and Unix (any Linux distribution) is required, but the instructor will give a quick review of advanced C and C++ programming aspects.

Course overlap: This course complements 6339: 6339 involves basic machine learning, tools and main techniques to analyze big data, from a high-level language (SQL, Python, R). In contrast, this new “Systems” course goes deeper into how memory, storage is managed and how ML/graph algorithms hre modified to scale to large data size and how they can work in parallel. The course has a minor overlap with 6339, 6340 and 6360. The main difference with 6339 is that there is more emphasis on internal systems aspects and little theory on machine learning. The main difference with 6340 (discontinued) is that all review/fundamental DB topics have been removed and programming is mostly C++; most fundamental DB topics from 6340 have been moved to 6339. From a programming standpoint the course requires knowing system-level programming in C, as taught in the undergrad OS course and knowing popular algorithms in C++ or Java, as taught in the undergrad algorithms course.

Instructor Background and Expertise

Brief academic bio (taken from my web page):

Education:

- B.Sc. in Applied Mathematics (Actuarial Science), UNAM University, Mexico, 1992.
- M.S. in Computer Science, UNAM University, Mexico, 1996.

- Ph.D. in Computer Science, Georgia Institute of Technology, USA, 2000.

Research goal: developing scalable, interoperable, parallel algorithms available as libraries and programs to analyze big data. In my lab we develop programs in C++, Python and SQL, mostly in Unix. I currently work on adapting and optimizing a broad spectrum of algorithms in machine learning and graphs to work in modern data science languages. I conduct research on both multi-node processing with distributed memory/storage, as well as multi-threaded processing on multi-core CPUs and GPUs. I work on many science and engineering problems, but most of my applied research has been in medicine and healthcare.

General policies

- All questions about grading, programming, exam difficulty will be answered in the 1st lecture.
- Programs will be carefully graded with 10 test cases, with varying degree of difficulty. Sample test cases will be posted before the due date.
- We will setup an agile message communication system (e.g. whatsapp, slack, discord), instead of email. We will make an effort to answer within 24 hours.
- Late submission of programs will incur a grade penalty; no extensions can be given. Late submissions beyond 3 days cannot be accepted.
- Source code quality: TAs and instructor will check code clarity, indentation, comments and overall modularity.
- Academic honesty: Source code will be automatically checked for plagiarism. Source code cannot be shared. Source code obtained from the Internet must be disclosed in documentation.
- Resubmission of programs with errors or incomplete requirements will incur a penalty between 10% and 20%, depending on the magnitude of changes.
- If for any reason a student cannot take the midterm exam on the assigned date, the makeup exam will be an oral exam (short questions) in the instructor office or lab.
- Attendance is not taken. Face to face lectures will not be recorded. Online lectures may be recorded.
- Disruptive noises during class are unacceptable (whispering to next student, phone calls, typing on laptop).
- Arriving late in the classroom is discouraged (i.e. more than 10 mins late).