COSC4397: Systems for AI and Big Data
Instructor: Carlos Ordonez

1 Overview

1.1 Proposed Catalog Description

Storage: hardware, layouts, file system. External algorithms and indexing data structures for Big Data. Adapting Query and Transaction Processing to Big Data. Parallel Multicore CPUs and GPUs. Parallel and Distributed Systems for neural networks, machine learning models and graphs.

1.2 Overlap with other courses

Minor overlap with COSC3320 on search and sorting algorithms, but extended to parallel procesing and secondary storage (disk, SSD). Small overlap with COSC3380 on the system side. Minimal with Data Science I since this course is about programming libraries rather than learning how to call them. This course is project-oriented, like Data Science II, but with a "systems" focus instead of application of AI in practical problems.

1.3 Prerrequisites

Prerreqs: COSC3380, COSC3360. Recommended: COSC3320 (advanced algorithms), COSC 3337 (Data Science I).

From a theory perspective, students should have a background on: discrete math, search/sort algorithms, time and space complexity analysis, database theory, data science.

Advanced programming: On the programming side, it is highly recommended students have "systems infrastructure" experience (memory management, I/O, networking) in C or C++ (in GNU C or C++ preferably, but Microsoft Visual C++ or C#, or Intel C++ OK) and experience in modern Unix (Linux or any proprietary Unix distribution). Experience with Python or Java helps, but it is insufficient. In general, operating systems and C-style programming have proven to be the most common weaknesses for students. Familiarity with modern Unix and C/C++ (e.g. Linux Ubuntu, Linux Redhat, GNU C++). However, to make the course self-contained the course will give a review of C++ systems libraries, fundamental aspects of each prerreq throughout the course.

1.4 Learning Objectives

Students taking this course will learn

- overview of parallel CPUs and GPUs for AI and its implication for systems programming
- brief theory foundation for big data, including architecures, parallel speedup and CAP theorem
- template algorithms for big data

- Storage: I/O bottlenecks, flash memory vs disk, file system libraries Partitioned, compressed storage; block-based I/O (records)
- relaxing ACID requirements
- Neural networks and Machine learning on large data sets
- Data structures to store tables, matrices, text and web data
- Modern text file formats to exchange data nowadays (CSV, JSON), displacing XML. Established Binary formats.
- parallel processing in multi-core and multinode architectures

2 Course Contents

This is a "systems" course (i.e. advanced C/C++, advanced Linux, advanced Python), learning how systems for AI ar large scale are programmed and optimized. This course will provide a solid foundation to develop software systems on parallel and distributed DBMSs and Big Data.

Textbooks: Selected chapters from textbooks [1, 2, 3].

The first part of the course (about one third) will cover theory that remains important today (surviving decades) and in the second part (two thirds) it will go into programming internals of modern systems to store, read data and analyze in a broad sense (a moving target with open-source and evolving hardware).

Theory: parallel sepeedup, I/O cost models, external search/sort algorithms, external indexing data structures, graphs, CAP theorem, algorithmic acceleration of machine learning algorithms, main neural network architectures and fwd/backward propagation

systems: New hardware: multicore CPUs, GPUs, large RAM, old HDD, SSD, NVM Data management: Buffer management (main memory addressing, segmentation, overriding OS page eviction algorithms, main memory databases), secondary storage manager (file systems, disk, solid state, row vs column storage, data partitioning), transactions (relaxed ACID,improvements over 2PL, optimistic, lock-free on multi-core, distributed transactions on partitioned files), Query processing (relational and non-relational): (sequential query plan and optimization, parallel query processing, physical database algorithms, fault tolerance for transaction processing (recovery), fault tolerance for query processing (parallel, incremental)). document storage and search, object (document) updating (CRUD) in NoSQL (queries without SQL, beyond SQL), big data storage (text, JSON, text, images, web), linear algebra, text pre-processing, text indexing, key-value stores, data frames, arrays, graphs. Scalable, Parallel Machine Learning: accelerating SGD, parallel matrix multiplication in CPUs and GPUs, data set partitioning, streaming data sets, fast aggregation, sparse matrices and tensors,

3 Grading

- Weights:
  80% programming project
  20% partial exam.
- One programming project mainly in C++ (and some pure C), with some Python, developed in 2 phases. A non-working program will get 10 points.
- Partial exam (no final exam): 10 questions, closed-electronics, in-class, written answer, grade 0-100. The exam is closed-everything: in-class, no notes, no textbook, no electronic devices. The exam will consist of 10 questions, where each question will require writing a short answer or a small fragment of source code (e.g. C++ or SQL). Regular semester: exam given around 10th week.

- Required attendance: 80% for face to face lectures and 90% for online lectures. Students below this threshold will be reported to UH.

Programming project (Regular semester): The project will require low-level systems programming in C++ (pointers in RAM, binary I/O, threads, maybe sockets) for transaction, query processing, neural networks on large files. Since many old libraries and Linux itself are written in C, students will have to tweak and program some parts in "classic C" or modern C++ (e.g C++ 17 or later). The project will be developed in 2 phases.

Source code development: The C++ source code will be developed in modern Unix (Linux), tested on a Linux server (with sufficient RAM and VCPUs for a database or AI workloads). The instructor will provide enough C and C++ code and libraries to get started (i.e. students are not expected to develop the C++ code from scratch). Programming projects will be developed in pairs (i.e. a team of 2 students) and they must run from the command line like traditional Unix tools (executable called with input parameters). Students will learn free open source code (FOSS) has copyright, with legal and academic implications.

Academic honesty: Sharing source code among teams is considered cheating and it will be reported to UH. Customizing and extending source code obtained from AI (LLMs like gemini, chatgpt, meta) and the Internet is acceptable, but it must be disclosed in the project documentation (LLM website, final prompts, exact website http address, exactly which C++/C functions or C++ classes were customized).

References

[1] H. Garcia-Molina, J.D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 1st edition, 2001.

[2] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2006.

[3] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 1st edition, 2001.