CSE 470|570: Introduction to Parallel and Distributed Processing

Course Information

- **Date/Time:** MWF, 4:10-5:00pm
- **Location:** Baldy 200G
- **Credits:** 3
- **CSE Focus Area:** Software and Information Systems (SW)

Instructor

Dr. Jaroslaw Zola

Department of Computer Science and Engineering

Email: jzola@buffalo.edu Web: http://www.jzola.org/ Twitter: @rzolau

For all email communication, please make sure to add prefix `[IntroPDP]` to mail subject.

Course Description

This course is intended for students interested in the efficient use of modern parallel systems ranging from multi-core processors and many-core accelerators to large-scale distributed memory clusters. The course puts equal emphasis on the theoretical foundations of parallel computing, and on practical aspects of different parallel programming models. It begins with a survey of common parallel architectures and types of parallelism, and then follows with an overview of formal approaches to assess scalability and efficiency of parallel algorithms and their implementations. In the second part, the course covers the most common and current parallel programming techniques and APIs, including for shared address space, many-core accelerators, distributed memory clusters and big data analytics platforms. Each component of the course involves solving practical computational and data driven problems, ranging from basic algorithms like sorting or searching, through numerical data analysis, to large graphs processing.

Course Organization

The course consists of a series of lectures organized into five topical modules. Each lecture module is complemented with a programming assignment exposing practical aspects of the covered material. Tentative course outline is provided below:

- Overview of parallel processing landscape: why and how, types of parallelism, Flynn's taxonomy and brief overview of parallel architectures, practical demonstration of CCR as an example HPC center. Basic concepts in parallel processing: formal definition of parallelism, concepts of work, speedup, efficiency, overhead, strong and weak scaling (Amdahl's law, Gustafson's law), practical considerations using parallel reduction and parallel prefix.
- Multi-core programming: shared memory and shared address space, data and task parallelism, summary of available APIs (OpenMP, TBB). Practical examples using OpenMP and parallel merge sort, pointer jumping, parallel BFS and basic linear algebra.
- Distributed memory programming: Message Passing Interface, latency + bandwidth model, distributed hashing, sample sort, parallel BFS, and basic linear algebra (matrix-vector, maxtrix-matrix products) with relation to graph algorithms.
- Higher-level programming models: stateless programming in Map/Reduce, Apache Spark and fault-tolerance via Resilient Distributed Datasets. Triangle counting, connected components, single source shortest path using Spark.
- Many-core programming: SIMD parallelism and massively parallel GPGPU accelerators. Brief overview of available APIs (OpenACC, oneAPI, OpenCL, SYCL). Programing GPUs in NVIDIA CUDA: data movement and organization, 1D/2D stencils, parallel prefix, matrix-matrix product.

Course Prerequisites

The course has no specific prerequisites for graduate students. For undergraduate students, CSE 220 “Systems Programming” and CSE 331 “Introduction to Algorithms” are prerequisites, as the course requires some experience in synthesis and analysis of algorithms, and programming close to the executing platform. The course has significant programming component, hence a rudimentary ability to learn new programming constructs is expected. Specifically, C++ (multi-core, many-core, MPI) and Python (Map/Reduce, Spark) will be used extensively during the course. Again, only basic ability to adopt to new programming languages is required and not expert knowledge of these languages.

Please note however that all students must:

1. **Have basic understanding of linear algebra and graph theory.**
2. **Be well versed in asymptotic notation, recursion and fundamentals of probability.**
3. **Be able to reason about complexity of algorithms.**
4. **Be able to interact with Linux/Unix systems using command line interface (e.g., tools like `ssh`, `rsync`, `tar`).**

Program Outcomes

Upon completion of this course you will:

- Gain basic understanding of fundamental concepts in parallel computing.
- Be able to identify and leverage common parallel computing patterns.
- Be able to properly assess efficiency and scalability of a parallel algorithm/application.
- Become proficient in using at least one parallel programming technique, and familiar with several others.

The course adopts the Student Outcomes from the Engineering Accreditation Commission of ABET (see here).

Course Learning Outcome	**Program Outcomes/Competencies**	**Instructional Method(s)**	**Assessment Method(s)**
Basic understanding of fundamental concepts in parallel computing	**1:** An ability to analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions	Lectures, assignments	Assignments, exams
Identifying and leveraging common parallel computing patterns	**2:** An ability to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline	Lectures, assignments	Assignments, exams
Properly assessing efficiency and scalability of a parallel algorithm/application	**6:** An ability to apply computer science theory and software development fundamentals to produce computing-based solutions.	Lectures, assignments	Assignments, exams
Proficiency in using at least one parallel programming technique, and familiarity with several others	**2:** An ability to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline	Lectures, assignments	Assignments, exams

Program Outcome support

Program Outcome	**1**	**2**	**3**	**4**	**5**	**6**
Support Level	3	3				3

Not Supported, 1: Minimally Supported, 2: Supported, 3: Strongly Supported

Course Requirements

The course has three requirements:

1. Midterm exam testing your understanding of a parallel computing landscape and basic concepts in parallel processing.
2. Programming assignments exposing you to the practical aspects of the covered material. There will be four assignments in total, one at the end of each course module (one after multi-core programming, one after distributed memory and so on). In each assignment, you will be asked to implement and benchmark a specific parallel algorithm relevant to real-life applications (using resources provided by CCR).
3. Final exam testing your overall understanding of the material.

Grading Policy

The final grade will be weighted average: 20% midterm exam, 30% final exam, 50% programming assignments. Exam and programming assignments will be the same for graduate and undergraduate students. However, criteria to decide the final grade will be different for graduate and undergraduate students. Specifically, the number-to-letter grade mapping will be done as indicated in the tables below.

Graduate

Score	Grade	Points
95-100	A	4.0
90-94	A-	3.67
80-89	B+	3.33
70-79	B	3.0
60-69	B-	2.67
55-59	C+	2.33
50-54	C	2.00
45-49	C-	1.67
40-44	D	1
0-39	F	0.0

Undergraduate

Score	Grade	Points
93-100	A	4.0
85-92	A-	3.67
75-84	B+	3.33
65-74	B	3.0
55-64	B-	2.67
50-54	C+	2.33
45-49	C	2.00
40-44	C-	1.67
35-39	D	1
0-34	F	0.0

In general, no incomplete grades (“IU” or “I”) will be given. However, in special circumstances that are truly beyond your control and justify incomplete grade, we will follow the university policy on incomplete grades, available here (for graduate students) and here (for undergraduate students).

Assignments Grading

There will be four programming assignments that will require that you develop a parallel code, test it on CCR resources, and report your findings. Assignments are graded using a combination of automated tools that test correctness and efficiency of your solution, and code review by a human to assess quality of the code. In general, the following main criteria are used for grading:

Criterion	**Comment**
Correctness	Code must compile/execute and return correct answers. If code does not compile/execute it is not graded. If it is entirely correct (i.e., passes coverage tests) no points are deducted. Otherwise (i.e., it fails for some cases) points are deduced accordingly. **Note also that your code must produce output exactly as specified in the assignments description**.
Performance	It is not enough that your code gives correct answer. Your code **must be efficient**, and use the best possible parallel programming techniques to achieve the goal. You have to pay attention to how you solve the problem. Bad decisions are penalized. After all, parallel computing is about performance!
Quality	**Your code cannot be a lousy last minute submission!** You have to pay attention to the details. One way to think about it is that your code must be of the quality that would warrant Pull Request in a larger project.

Course Materials

This course does not rely on one specific textbook. The following books are suggested but not required:

- M. McCool, J. Reinders, A. Robison, “Structured Parallel Programming: Patterns for Efficient Computation,” Morgan Kaufmann, 2012, ISBN-13: 9780124159938.
- A. Grama, G. Karypis, V. Kumar, A. Gupta, “Introduction to Parallel Computing (2nd Ed.),” Pearson, 2003, ISBN-13: 9780201648652.

- H. Karau, A. Konwinski, P. Wendell, M. Zaharia, “Learning Spark: Lightning-Fast Big Data Analysis (2nd Ed.),” O’Reilly Media, 2020, ISBN-13: 9781492050049.
- D.B. Kirk, W.W. Hwu, “Programming Massively Parallel Processors: A Hands-on Approach (3rd Ed.),” Morgan Kaufmann, 2010, ISBN-13: 978-0128119860.

Before ordering any of these books, please contact your instructor regarding books availability. Additionally, several research papers may be referenced during the course.

Significant online resources, including tutorials, API documentations, “quick start guides”, etc. will be referenced during the course, and can be easily found using your favorite search engine.

Computing Resources

For the duration of the course you will be granted access to the resources (including storage) provided by the UB Center for Computational Research (CCR). CCR is the state-of-the-art HPC and data center hosting clusters, multi-core compute nodes, and compute nodes with GPGPU accelerators. It provides programming and execution environments supporting all types of parallelism covered in this course.

Additionally, a virtual environment with a Linux distribution and all required compilers and runtime systems will be available for offline use.

Academic Integrity

Academic integrity is critical to the learning process. It is your responsibility as a student to complete your work in an honest fashion, upholding the expectations your individual instructors have for you in this regard. The ultimate goal is to ensure that you learn the content in your courses in accordance with UB’s academic integrity principles, regardless of whether instruction is in-person or remote.

You must be familiar with the university and departmental policies on academic integrity! The university policies are available from https://www.buffalo.edu/academic-integrity/policies.html. The CSE policies are available from the CSE web page.

Any violation of these policies, including but not limited to cheating on any course deliverable (e.g., homework project, exam, etc.), will result in automatic failure of the course. There will be no leniency! If you decide to use a code from some external source, e.g., an open source project, you must include a proper and clearly visible attribution in your product (it is a good idea to contact your instructor to check if the code you plan to use is admissible).

Thank you for upholding your own personal integrity and ensuring UB's tradition of academic excellence.

Accessibility Resources

If you have any disability which requires reasonable accommodations to enable you to participate in this course, please contact the Office of Accessibility Resources in 60 Capen Hall, 716-645-2608 and also the instructor of this course during the first week of class. The office will provide you with information and review appropriate arrangements for reasonable accommodations, which can be found on the web at: http://www.buffalo.edu/studentlife/who-we-are/departments/accessibility.html.

Counseling Service

As a student you may experience a range of issues that can cause barriers to learning or reduce your ability to participate in daily activities. These might include strained relationships, anxiety, high levels of stress, alcohol/drug problems, feeling down, health concerns, or unwanted sexual experiences. Counseling, Health Services and Health Promotion are here to help with these or other issues you may experience. You learn can more about these programs and services by contacting:

Counseling Services

- 120 Richmond Quad (North Campus), 716-645-2720
- 202 Michael Hall (South Campus), 716-829-5800
- https://www.buffalo.edu/studentlife/who-we-are/departments/counseling.html

Health Services

- Michael Hall (South Campus), 716-829-3316
- https://www.buffalo.edu/studentlife/who-we-are/departments/health.html

Office of Health Promotion

- 114 Student Union (North Campus), 716-645-2837
- https://www.buffalo.edu/studentlife/who-we-are/departments/health-promotion.html.

Sexual Violence

UB is committed to providing a safe learning environment free of all forms of discrimination and sexual harassment, including sexual assault, domestic and dating violence and stalking. If you have experienced gender-based violence (intimate partner violence, attempted or completed sexual assault, harassment, coercion, stalking, etc.), UB has resources to help. This includes academic accommodations, health and counseling services, housing accommodations, helping with legal protective orders, and assistance with reporting the incident to police or other UB officials if you so choose. Please contact UB's Title IX Coordinator at 716-645-2266 for more information. For confidential assistance, you may also contact a Crisis Services Campus Advocate at 716-796-4399.

Please be aware UB faculty are mandated to report violence or harassment on the basis of sex or gender. This means that if you tell me about a situation, I will need to report it to the Office of Equity, Diversity and Inclusion. You will still have options about how the situation will be handled, including whether or not you wish to pursue a formal complaint. Please know that if you do not wish to have UB proceed with an investigation, your request will be honored unless UB's failure to act does not adequately mitigate the risk of harm to you or other members of the university community. You also have the option of speaking with trained counselors who can maintain complete confidentiality. UB's Options for Confidentially Disclosing Sexual Violence provides a full explanation of the resources available, as well as contact information. You may call UB's Office of Equity, Diversity and Inclusion at 716-645-2266 for more information, and you have the option of calling that office anonymously if you would prefer not to disclose your identity.