CS406 / CS531 – Parallel Computing – Spring 2023-2024
(3 credits)

Prerequisite: CS301

Learning objectives

- At the end of the course the student will understand the evolution of high performance and parallel computing with respect to laws and modern hardware.
- At the end of the course the student will be competent to measure the performance of parallel and distributed programs.
- At the end of the course the student will be competent to evaluate whether a parallel and distributed application is efficient or not and propose techniques to increase the performance.
- At the end of the course the student will be able to design, apply, and analyze parallel algorithms in problem solving.
- At the end of the course the student will gain hands-on experience with GPU programming with CUDA and multicore programming with OpenMP.

Topics covered:

❑ **Introduction to (high-performance, parallel, etc.) Computing (2 weeks)**
a. **Computing** – *History, Turing Machines von Neumann architecture*
b. **Implicit Parallelism**: *Trends in microprocessor architectures - Superscalar execution, branch prediction*
c. **Limitations of Memory System Performance** – *Measuring system performance*, *memory wall, latency, bandwidth, caches (direct mapping, set associativity), temporal/spatial locality, multithreading, prefetching*
d. **Dichotomy of Parallel Computing Platforms** - *Control unit, SIMD, MIMD, vectorized instructions (SSE, AVX)*
e. **Communication Costs in Parallel Machines** - *Shared address space, message passing, NUMA and UMA architectures, interconnection networks*
f. **Messaging Cost Models and Routing Mechanisms** - *Store and forward routing, packet routing, cut-through routing, congestion, rules of thumb for communication*

❑ **Principles of Parallel Algorithm Design (1 week)**
a. **Tasks and Dependency Graphs** - *Granularity, degree of concurrency, critical path length*
b. **Task Interaction Graphs** - *Processes and mapping, partitioning, characteristics of task interactions*
c. **Decomposition** - *Recursive, data, exploratory, speculative, owner-computes rule*
d. **Mapping** - *Static and dynamic mapping, maximizing data locality, minimizing congestion, overlapping communication and computation.*

❑ **Analytical Modeling of Parallel Performance: (1 week):** Performance metrics for parallel systems Scalability of parallel systems Cost optimality, scalability, efficiency Asymptotic isoefficiency Minimum execution time, Other scalability metrics —memory and time constrained scaling —serial fraction —Amdahl's law

❑ **Programming Shared-Memory Machines with OpenMP (2.5 weeks)**
a. **Multithreading –** *Concurrency, parallelism, shared memory computers, false sharing*
b. **Synchronization primitives –** *OpenMP memory model. barrier, locks, atomic operations, critical regions*
c. **Worksharing –** *Loops, static/dynamic/guided scheduling, reduction, shared/private variables,*
d. **OpenMP Tasking –** *Scheduling and dependencies*
e. **Thread binding and Memory Placement –** *OpenMP memory model, system topology, affinity and NUMA.*

❑ **Programming GPUs with CUDA (3.5 weeks)**
a. **Simple GPU Usage:** *CUDA software stack, thread hierarchy (thread, block, grid), CUDA memory model, calling kernels, compilation, nvprof, cuda-memcheck*
b. **Synchronization in CUDA:** *Cooperating threads, warps, block synchronization, global synchronization, data races, atomic operations*
c. **Memory model:** *Using global memory, array of structures, structure of arrays, coalesced access, hiding latency, configuring and programming with shared memory, shared-memory banks and bank conflicts.*
d. **GPU occupancy***: Resource limits, registers, shared memory*
e. **Unified memory:** *Zero-copy memory, pinned memory, mapping, on-demand paging, oversubscription.*
f. **CUDA Streams:** *Pipelining, overlapping, advanced scheduling, creating streams, pinned memory.*

❑ **Collective Communication Operations (1 week):** *One-to-all broadcast and all-to-one reduction. all-to-all broadcast and reduction. all-reduce and prefix-sum operations, scatter and gather, all-to-all personalized communication, circular shift, optimizing collective patterns.*

- **Algorithms for Dense Matrices, Vectors, and Arrays (1 week):** *Matrix-vector multiplication, 1D and 2D partitionings, matrix-matrix multiplication, 2D partitioning and Cannon's algorithm, communication bounds, 2.5D matrix multiplication.*

- **Algorithms for Sparse Matrices, Tensors and Graphs (if time permits, 1 week):** *BFS, SpMV, Triangle Counting, Connected Component Decomposition...*

Instructor

Dr. Kamer Kaya, FENS G012, ext. 9566, kaya@sabanciuniv.edu
Assistants: Information (offices, office hours, etc.) will be available through SUCourse.

Lecture Hours:

- Mon, 08:30-10:30, FASS G006
- Wed, 08:30-09:30, FENS G035

Textbook(s): There are no textbooks, slides are the main material
Reference books are

- Introduction to Parallel Computing, Second Edition. Ananth Grama. Anshul Gupta. George Karypis. Vipin Kumar.
- Programming Massively Parallel Processors: A Hands-on Approach, David B. Kirk and Wen-mei W. Hwu.

Reminder: we will not stick to the textbooks; you are responsible material covered in class. Thus it is important to attend to classes. This being said, attendance will not be used in grading.

Tentative Grading (subject to change)

- No Midterms or Finals (0%)
- Homework assignments (50%) – There will be 5 homework assignments which are not of equal weight. Homework grading will mostly be based on correctness, as well as the performance of the execution. No debugging will be done during grading.
- Project (50%) – The problem(s) will be announced later – see the explanations at the end of the document.

Other Rules and Remarks

- Weighted average is not the only criterion in letter grading; ratio of the HW scores to project scores may also be taken into consideration in case of uncorrelated grades.
- Since there is no midterm or final, there is no makeup.
- **You may be asked to take an oral exam if the instructor of the course thinks that it is necessary.**

See course website for other, but important, details

Plagiarism, Homework Trading and Cheating will not be tolerated

Project Assignment Instructions: The final project is to solve a problem with parallel processing on **gandalf** or **nebula** (you will have the accounts during the term). You will work on a problem, analyze its compute scaling requirement, collect the data, design and implement a parallel software that can run both on multicore CPUs and manycore GPUs, and demonstrate scaled performance of an end-to-end application.

Your submission will be evaluated on various problem sizes and thread counts to demonstrate both weak and strong scaling using appropriate metrics (run-time, throughput, efficiency, ...).

Project Topic: The project title is **Computing Sparse Matrix Permanents with OpenMP, CUDA and MPI**. See [1] for the description of permanent and its applications. See [2] for a recent paper on their parallel computation. In this project, you will be focused on sparse matrices only.

[1] https://en.wikipedia.org/wiki/Permanent_(mathematics)

[2] https://www.diva-portal.org/smash/get/diva2:1640481/FULLTEXT01.pdf

Group Size: Students are required to form teams and to partition the work among the team members. The project must be done in teams with 2 students each (**if the number of students is odd, there will be one single student team**). If you are single, you can use SUCourse forum to find prospective team members. We reserve the right to assign different grades to each group member if it becomes apparent that one of them put in a vastly different amount of effort than the other.

Project Milestones: There are five milestones for your final project. It is critical to note that no extensions will be given for any of these milestones for any reason. Projects submitted after the final due date will not be graded.

- **Progress report** 05/04/2024
- **Presentation** (to the instructor) 01/06/2024 – 10/06/2024
- **Final report (and other deliverables, e.g., source codes)** 10/06/2024

Progress Report: Your group needs to submit a project progress report in PDF covering the main aspects in the design and implementation of the parallel application with the following information (the detail level depends on the progress).

- What is the problem you are trying to solve with this application?
- A basic literature survey.
- Describe the complexity of your chosen/novel algorithm(s).
- The levels of parallelism exploited, the types of parallelism (mainly OpenMP).
- Provide an analysis/profiling of the existing sequential/parallel code (for each of the above).
- Describe the main overheads (memory access, communication, synchronization, load balancing etc.) in the parallelization and techniques that are applied to mitigate them.
- Estimate the theoretical speed-up and scalability expected you achieved until now.

Presentation, Final Report and Codes: Final presentation will be given during the final exam week to the instructor. Along with your final report, the final deliverables must cover.

- The levels of parallelism exploited, the types of parallelism (OpenMP, CUDA and MPI).
- Final, detailed performance evaluation (speed-up, throughput, weak and strong scaling) and discussion about overheads and the impact of each optimization performed.
- Final discussion about goals achieved, improvements suggested, lessons learnt, future work, interesting insight.
- Citations to the literature.
- Software with evaluation data sets, test cases.

Project Software: If we cannot run your application from the instructions included with your submission, we will not be able to grade this portion of your project. Your performance results should be reproducible, so you should provide all the information of the system and the environment needed to reproduce your tests.

Project Presentations: You will have 25 minutes to briefly present your project followed by 5 minutes of discussion time. You may prepare 3-4 slides for problem introduction, but we will enforce the 25-minute time limit. Focus on your

accuracy/performance. Answer questions such as what insights did you gain? What is the two most important optimizations you would like to present? Upload the presentation slides to SUCourse.

Project will be graded on the depth of work undertaken and communication (web site, presentation):

- 30%: Project progress report
- 70%: Final documentation, presentation, and software source code

Extra points may be earned for the use of **advanced features** like:

- Using multiple GPUs/nodes at once.
- Challenging parallelization or implementation aspects.

Final Testing:

- Install a zip file for the codes that has a README file.

- Have a script "**compile.sh**" on the folder that generates all the executables.
  - We will use **nebula** so try to optimize your codes based on the CPU(s)/GPU(s) we have there.

- For each executable, this command should work.
  - **./executable matrix_file_name no_threads/no_GPUs**

- The executable must first output the permanent (can be in scientific format) and execution time (in seconds) separated by a tab. For instance
  - **2403738 5.5**

- The n x n matrix files will be given in this format.

$$
\begin{aligned}
&n\ m\\
&i_1\ j_1\ val_1\\
&i_2\ j_2\ val_2\\
&i_3\ j_3\ val_3\\
&\ldots.\\
&i_m\ j_m\ val_m
\end{aligned}
$$

  - The first row in the file shows the number of matrix rows/columns and then number of nonzeros inside the matrix.
  - Then each next row of the file corresponds to a nonzero inside the matrix.
    - i and j values are integers – start from 0 and at max they can be n-1.
    - each val is a real number.