**Heterogenous Computing Projects**

**Overview**:

Project ideas will be released first week of the course and students must form group to start working on the projects. The group project should be implemented in C/C++, and then use a parallel programming model such as SYCL and DPC++ to speed up the code. This project will be done in teams of 2 or 3 people. The final deliverable will be a report and the code (with clear instructions for running it).

|  |  |
| --- | --- |
| **Milestones** | **Due** |
| Form groups | **1 week** |
| |  | | --- | | Project description | | **1 week** |
| |  | | --- | | Interim report | | **3 weeks** |
| |  | | --- | | Project presentations | | **3 weeks** |
| |  | | --- | | Final project and report | | **2 weeks** |

**Project phases:**

Each project has 4 phases that students must follow to be successful:

Phase 1: Understanding Device Characteristics for Heterogeneous Computing and Justifying Device Selection

The goal of this phase is to ensure students understand the different characteristics and advantages of CPUs, GPUs, and FPGAs. Students will research and document the specifics of each type of hardware, including architecture, computation capacity, memory hierarchy, and typical use cases. Then, given a set of example tasks or applications (e.g., image processing, machine learning, complex calculations), students will justify which device they would choose for each task and why, demonstrating their understanding of heterogeneous computing.

Phase2: Exploring Parallel Patterns and Performance Bottlenecks in Heterogeneous Computing

Students will start with a basic serial program (e.g., matrix multiplication, a sorting algorithm, or a simple machine learning algorithm) and modify it to run in parallel on different devices (CPU, GPU, and FPGA). They should document their process and the challenges they face, including synchronization, memory access patterns, etc. Using the tools available in Intel DevCloud, they should also perform performance analysis to identify and discuss bottlenecks.

Phase 3: Harnessing the Power of SYCL and DPC++ for Heterogeneous Programming

This phase will give students practical experience with SYCL and DPC++. They will take the optimized version of their parallel program and rewrite it using these languages. They should compare the performance of their new program with the previous versions and discuss the benefits and potential drawbacks of using SYCL and DPC++.

Phase 4: Performance Tuning and Optimization in Heterogeneous Computing Systems

Building on phase 3, students will now focus on performance optimization. They will experiment with different techniques such as optimizing memory usage, thread management, loop unrolling etc., to improve the performance of their parallel program. They should discuss why they selected each technique and how it improves performance.

Each phase builds on the last, allowing students to progress from understanding the basics of heterogeneous computing systems to hands-on experience with optimization and programming in SYCL and DPC++. They will finish with a thorough understanding of heterogeneous computing and practical experience in optimizing and programming these systems.

**Five major projects are considered as follow:**

1- Parallel Implementation of Sequential Minimal Optimization (SMO) Algorithm and Model Selection for Support Vector Machines

2- A Parallel Hybrid Framework for Graph Processing

3- Distributed Learning for Deep Neural Networks

4- Parallel Simulation of Information Diffusion on Large Social Network Graphs

5- Parallel Patch Matching for Image Segmentation

Project with Bonus: Reproduce results from a paper, extend to current systems - e.g., CPU vs. GPU paper (pick a small number of application kernels)

**Project 1: Parallel Implementation of Sequential Minimal Optimization (SMO) Algorithm and Model Selection for Support Vector Machines**

\*\*Project Description:\*\* The Sequential Minimal Optimization (SMO) algorithm is widely used for training Support Vector Machines (SVM), a critical algorithm in machine learning. However, SMO is inherently sequential and thus a prime candidate for parallelization. The students will implement the SMO algorithm, then parallelize it using SYCL and DPC++. They will then use this parallelized version for model selection (by grid search or random search) over a large parameter space.

\*\*Deliverables:\*\*

- The initial SMO algorithm implementation and its parallelized version.

- A report comparing the performance of the parallelized and non-parallelized versions of the algorithm, including an analysis of the speedup and efficiency.

- A SVM model selection procedure using the parallelized SMO.

Suggestion:

For this project, students could work with standard datasets available in UCI Machine Learning Repository or similar sources. Some of these datasets could be:

Iris Dataset: A simple multivariate dataset that could be used for a binary or multi-class classification task.

Breast Cancer Wisconsin (Diagnostic) Dataset: This dataset can be used for a binary classification task, and it's slightly more complex than the Iris dataset.

Wine Dataset: A multi-class classification problem where the classes are ordered but not balanced (e.g., there are many more normal wines than excellent or poor ones).

Please note that the SVM is a binary classifier; for multi-class tasks, one would typically use a One-vs-One or One-vs-All strategy.

**Project 2: A Parallel Hybrid Framework for Graph Processing**

\*\*Project Description:\*\* Graph processing is a crucial part of many modern applications, from social networks to biological networks. However, efficient graph processing on heterogeneous systems is a challenging task. In this project, students will design and implement a hybrid parallel graph processing framework using SYCL and DPC++ that can efficiently utilize CPUs, GPUs, and FPGAs.

\*\*Deliverables:\*\*

- The hybrid parallel graph processing framework.

- A report detailing the design choices made during the development of the framework, along with an analysis of how the framework utilizes the heterogeneous system.

- Demonstrations of the framework's use in processing real-world graph datasets.

Suggestion**:**

You can consider the following social network graphs:

Stanford Large Network Dataset Collection (SNAP): This dataset contains a variety of large social network graphs, including social networks, web graphs, and road networks. Particularly relevant might be the Twitter, Facebook, or YouTube networks.

DBLP Computer Science Bibliography: This dataset provides a comprehensive list of papers, authors, and the citations among them, and can be modeled as a graph with authors as nodes and co-authorships as edges.

Amazon product co-purchasing network: This dataset includes metadata and reviews for a subset of Amazon products, and can be represented as a graph with products as nodes and co-purchasing relationships as edges.

**Project 3: Distributed Learning for Deep Neural Networks**

\*\*Project Description:\*\* The process of training deep neural networks can be computationally intensive. Heterogeneous systems, with their variety of compute resources, offer the potential for improved performance. In this project, students will implement a simple deep learning model (e.g., a multi-layer perceptron or convolutional neural network) and design a distributed learning approach using SYCL and DPC++.

\*\*Deliverables:\*\*

- The deep learning model and its distributed learning version.

- A report comparing the performance of the distributed and non-distributed versions of the model training, including an analysis of the speedup and efficiency.

- Application of the distributed learning approach on a benchmark dataset (like MNIST or CIFAR).

Suggestion:

For deep learning tasks, the choice of network and dataset often depends on the specific task at hand. However, I can suggest the following commonly used network architectures and corresponding datasets:

Convolutional Neural Network (CNN) on CIFAR-10 or CIFAR-100: These are datasets of 50,000 32x32 color training images and 10,000 test images, labeled over 10 or 100 categories, respectively. A CNN is a good fit for image classification tasks.

Multilayer Perceptron (MLP) on MNIST or Fashion-MNIST: MNIST is a dataset of handwritten digits and Fashion-MNIST is a dataset of Zalando's article images. Both datasets have a training set of 60,000 examples and a test set of 10,000 examples. An MLP is a good fit for these image classification tasks.

Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM) on IMDB Review Dataset: The IMDB Review Dataset is a set of 50,000 highly polar movie reviews for binary sentiment classification. RNNs or LSTMs are good fits for natural language processing tasks like sentiment analysis.

**Project 4: Parallel Simulation of Information Diffusion on Large Social Network Graphs**

\*\*Project Description:\*\* Understanding how information diffuses in a network is vital for a range of fields, including epidemiology, viral marketing, and more. In this project, students will implement a model of information diffusion (like the Independent Cascade Model or Linear Threshold Model), then parallelize this simulation using SYCL and DPC++. The simulations should be conducted on large real-world social network datasets.

\*\*Deliverables:\*\*

- The information diffusion model and its parallelized version.

- A report comparing the performance of the parallelized and non-parallelized versions of the simulation, including an analysis of the speedup and efficiency.

- Results of the simulations on real-world social network datasets, along with a discussion of the findings.

Suggestion:

Students could use the following social networks:

Facebook Large Page-Page Network: This dataset consists of 'like' relationships between pages on Facebook. Nodes represent official Facebook pages while the edges are 'like' links.

Twitter Social Network: This graph was collected from Twitter dataset and it is a network of Twitter users. A directed edge in the network means that a user is followed by another user.

Reddit Hyperlink Network: This dataset includes all the post-to-post and post-to-subreddit links for all the posts made on the Reddit platform in 2014.

The model of information diffusion could be based on common models in the literature, such as the Independent Cascade Model or the Linear Threshold Model.

**Project 5: Parallel Patch Matching for Image Segmentation**

\*\*Project Description:\*\* Patch matching is a critical process in many image analysis tasks, including image segmentation, where the goal is to divide an image into segments that have a similar meaning. In this project, students will implement a patch matching algorithm for image segmentation, then parallelize this algorithm using SYCL and DPC++. They will test their implementation on real-world image datasets.

\*\*Deliverables:\*\*

The patch matching algorithm and its parallelized version.

A report comparing the performance of the parallelized and non-parallelized versions of the algorithm, including an analysis of the speedup and efficiency.

Results of the image segmentation on real-world image datasets, along with a discussion of the findings.

Suggestion:

Students can work with common benchmark datasets in image segmentation such as:

BSDS500: This dataset from the Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500) includes 500 natural images, along with hand-drawn, ground-truth segmentations.

PASCAL VOC: The PASCAL Visual Object Classes (VOC) datasets consist of images with object class annotations, including segmentations.

MS COCO: Microsoft Common Objects in Context (MS COCO) is a large-scale dataset for multiple computer vision tasks, including segmentation. The images in COCO are more complex and have more instances per image, making it more challenging than the other two.

***General Guideline for All Projects***:

*Research & Planning*: Understand tasks and how these tasks can be parallelized. Plan how to design a framework that can efficiently utilize CPU, GPU, and FPGA resources.

*Sequential Implementation*: Implement the algorithm in a sequential manner first. Make sure to validate this implementation with test cases.

*Parallel Implementation*: Using your knowledge of the SYCL and DPC++ languages, modify the sequential version of the algorithm to run in parallel. Make sure to utilize the different hardware accelerators available.

*Testing & Evaluation*: Test the framework on various processing tasks using real-world datasets. Evaluate the efficiency of the framework and how effectively it utilizes the available resources.

*Documentation & Reporting*: Document the design and implementation process of the framework. Include a detailed performance evaluation and explain the design choices made during the development of the framework. Document the entire process, including the challenges faced during the parallelization of the algorithm, performance comparisons, and the achieved results.