Project plan for degree project

Template version 1.6 – June 12, 2020

DV1478: BACHELOR THESIS IN COMPUTER SCIENCE

April 9, 2021

|  |  |  |
| --- | --- | --- |
| **Title** | Optimization of Heterogeneous Parallel Computing using Machine Learning | |
| **Classification** | Matrix Multiplication, Machine Learning, Optimization of Heterogeneous and Parallel Computing | |
| **Student 1** | **Name** | Devi Abhiseshu Adurti |
| **E-Mail** | dead20@student.bth.se |
| **Person nr** | 991120-T052 |
| **Program** | Bachelor Qualification Plan in Computer Science |
| **Student 2** | **Name** | Mohit Battu |
| **E-Mail** | mobt20@student.bth.se |
| **Person nr** | 991007-T175 |
| **Program** | Bachelor Qualification Plan in Computer Science |
| **Supervisor** | **Name and Title** | Suejb Memeti, Professor |
| **E-Mail** | suejb.memeti@bth.se |
| **Department** | Department of Computer Science |

# **Introduction**

A heterogeneous system is a combination of CPUs and GPUs that reduces communication latency, energy consumptions and provides increased performance by running multiple devices in a parallel mode[1]. It utilizes and combines the different types of resources available in our computer system**.** The main challenges of heterogeneous computing are workload partitioning that iswhen a system is trying to run heavy processes in the background and meanwhile there are simultaneous I/O (Input and Output) interruptions given by the user then the system by default assigns the workload to the CPU for handling the tasks which in turn creates a pressure on the CPU cores that might end up in a deadlock situation. It results in the consumption of all the available resources thereby reducing the performance of a computer system.

The system runtime needs to be equipped with an Intelligent learning model which could be able to foresight the upcoming problem and solve it by addressing the workloads to the respective CPUs and GPUs according to their availability. We have chosen a machine learning approach for this project to optimize the efficiency of a parallel computing system. The selected system-level benchmark is Matrix Multiplication which is a core for processing and identifying the patterns in an image.

There are many kinds of benchmarks like sorting, backpropagation, and depth-first search[2],[3]. A particular benchmark has various implementations of heterogeneous parallel computing applications based on GPU and CPU. The application here is referred to as an algorithm involved in performing a Matrix Multiplication. These applications are fed by the different input sizes and few of these applications perform better on the CPU for small input size and GPUs are preferred more for handling large input size. In this thesis, we focus on using a machine Learning model to determine the optimal selection of the algorithm based on the CPU and GPU execution time[4].

The Nvidia toolkit provides the means for writing cuda applications[5]. The matrix multiplication applications are run using the Nvidia toolkit to generate the execution times of CPU and GPU for multiple input configurations. A corresponding dataset for each algorithm is generated and These datasets are merged to get a single dataset containing execution time for all the matrix multiplication implementation. An ML model is trained and tested against a set of different input values to these algorithms. Which in return provides the optimal solution i.e., best algorithm for that task.

The developed model anticipates and provides a suggestion to the system that this task needs to be handled by the CPUs or GPUs. It boosts the overall efficiency of a system thereby releasing the computer resources which could be carried forward for other system operations.

# **Aims and Objectives.**

The aim of the thesis is the optimization of heterogeneous parallel computing algorithms for matrix multiplication. The algorithms taken here for matrix multiplication are naive, tiled, and square-filling algorithms. The matrix multiplication algorithms are integrated with a machine learning model to predict the performance of the algorithms on CPU and GPU such that the optimal technique is chosen during the system runtime.

Other objectives:

* To generate datasets of each matrix multiplication application concerning the CPU and GPUs.
* To structure data on the generated dataset and perform data pre-processing.
* To execute the exploratory data analysis on the refined dataset.
* To Select and Train the machine learning model with the generated dataset.
* To predict the optimal implementation and evaluate the performance of the model.

# **Research questions**

1. How does the different machine learning model affect the performance of the system in predicting the right execution time of the given input size of a matrix application?
2. How does the execution time predict by the model compares to the actual execution time of the algorithm?

# **Method**

The Nvidia toolkit facilitates developers for programming the cuda applications. Different types of Input sizes are fed into the matrix applications which generates the execution times of CPU and GPU. We build three different types of datasets for the applications of tiled, naïve, and square filling multiplication. These three different types of datasets are combined to get a single dataset containing execution time for all the matrix multiplication implementation. The generated dataset is structured and performed data pre-processing on it by using the panda’s library[6].

On to the refined dataset, mat plot and seaborn libraries are used for exploring the data analysis[7]. The different applications of the generated dataset are combined and trained by using the different types of a machine learning model. All the implemented machine learning models are then used to display the accuracy score of execution time such that the best performing model could be chosen. Further, the performance of the model is evaluated, and parallel tuning is performed to boost the performance. Finally, it will be used to predict the optimal algorithm for provided matrix size input.

# **Expected Outcomes**

The expected outcome of the project is to optimize the efficiency of matrix multiplication parallel computing algorithms by using machine learning models. Machine learning models are used to predict the type of algorithm to be implemented and predict the execution time taken in the CPUs and GPUs[8]. Depending on the volume of the data model assigns the workload to the CPU cores or the GPU cores thereby boosting the overall efficiency of a system and reduces the excess usage of computing resources thus freeing up more memory of a ram for other major operations to be performed in a system.

# **Time and Activity plan**

The Project is estimated to be done in 450 hours of work approximately. The work is seen to be done in consecutive steps such as Planning, Research, Modelling, Execution, Documentation, and Presentation. An estimated 45 hours per week is required to complete the project.

Planning - 50 hours

Research - 80 hours

Modelling - 110 hours

Execution - 130 hours

Documentation & Presentation - 80 hours.

|  |  |  |
| --- | --- | --- |
| Week | Activity | Details |
| Week 1 | Planning for topic | The thesis project topic is searched about the previous works. |
| Week 1 | Planning for Thesis  proposal | The project proposal first draft is prepared for the selected project topic. |
| Week 1-2 | Planning for final thesis proposal | The final project plan preparation and sending to the supervisor. |
| Week 2 | Planning | Making changes to Project Plan based on feedback received from the supervisor and submitting the Project Plan. |
| Week 2-3 | Research with literature review | Start working on the project by gathering information related to the research work for the project. |
| Week 3 | Research for method implementation | Gathering information and knowledge required for the implementation of the algorithms and models that be applied in the project. |
| Week 4-6 | Modelling | Modelling the solution using appropriate procedures and methods obtained from the research. |
| Week 6-8 | Execution of the solution | Executing the solution obtained from modelling and making required changes to the solution based on results received. |
| Week 8 | Evaluation of the solution. | Verifying the obtained results and again executing if changes are needed. |
| Week 9-10 | Documentation and Revision for the presentation. | Documenting the results about research questions and objectives. Preparing for the presentation. |

# **Limitations and risk management**

The matrix multiplication algorithm data set is generated from only one system, it is not considering the different systems GPU and CPU performance as a factor in determining the efficient optimal algorithm.

Some of the matrix multiplication algorithms are only limited to the square matrices’ multiplication. So, for non-square matrix sizes input, the efficiency would be limited to two algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Plan of Action** |
| Inaccuracy in results | Moderate | High | Reconstruction of algorithms and building a more efficient dataset. |
| Selection of appropriate model | Moderate | High | Different types of the model need to be tested against the dataset and an optimal is chosen. |
| Dataset avail-  ability | Moderate | Moderate | Generating a dataset concerning the model learning approach. |
| Dataset Pre-Processing | Low | Moderate | A raw dataset should be properly structured and cleaned. |

# **8 Supervisor Plan**

The supervision of the thesis is decided to have us providing the weekly progress of the project to the supervisor and whenever it is necessary a meeting with the supervisor will be conducted on demand.

# **References**

[1] H. Kang, H. C. Kwon, and D. Kim, “HPMaX: heterogeneous parallel matrix multiplication using CPUs and GPUs,” *Computing*, vol. 102, no. 12, pp. 2607–2631, Dec. 2020, doi: 10.1007/s00607-020-00846-1.

[2] K. Thearling and S. Smith, “An improved supercomputer sorting benchmark,” in *Proceedings Supercomputing ’92*, Minneapolis, MN, USA, 1992, pp. 14–19, doi: 10.1109/SUPERC.1992.236714.

[3] R. Murphy, J. Berry, W. McLendon, B. Hendrickson, D. Gregor, and A. Lumsdaine, “DFS: A Simple to Write Yet Difficult to Execute Benchmark,” in *2006 IEEE International Symposium on Workload Characterization*, Oct. 2006, pp. 175–177, doi: 10.1109/IISWC.2006.302741.

[4] T. Chen *et al.*, “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems,” *arXiv:1512.01274 [cs]*, Dec. 2015, Accessed: Apr. 05, 2021. [Online]. Available: http://arxiv.org/abs/1512.01274.

[5] S. Cook, *CUDA Programming: A Developer’s Guide to Parallel Computing with GPUs*. Newnes, 2012.

[6] D. Chen and Y. Yang, “Attribute Reduction for Heterogeneous Data Based on the Combination of Classical and Fuzzy Rough Set Models,” *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 5, pp. 1325–1334, Oct. 2014, doi: 10.1109/TFUZZ.2013.2291570.

[7] “IEEE Xplore Full Text PDF.” Accessed: Apr. 05, 2021. [Online]. Available: http://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=1311948&ref=aHR0cHM6Ly9pZWVleHBsb3JlLmllZWUub3JnL3N0YW1wL3N0YW1wLmpzcD9hcm51bWJlcj0xMzExOTQ4.

[8] J. Masek, R. Burget, L. Povoda, and M. K. Dutta, “Multi–GPU Implementation of Machine Learning Algorithm using CUDA and OpenCL,” *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 5, no. 2, Art. no. 2, Jun. 2016, doi: 10.11601/ijates.v5i2.142.