Milestone Report for "Miracle-Gro: Parallelizable Implementation of Random Forests in OpenMP"

Meher Mankikar (mmankika) and Deep Patel (dmpatel)

March 2023

1 URL

Miracle-Gro Project Page
Raw URL: https://dinodeep.github.io/15418-Project-Miracle-Gro/

2 Schedule and Future Plans

Below is the long-term schedule that was set by our initial proposal that we have been attempting to follow for the duration of the project.

Goal	Due Date	Status
Start Random Forest Implementation C++	4/8	Complete
Finish Sequential Implementation	4/15	Complete
Finish Profiling	4/19	In-Progress
Finish Parallelization	4/22	In-Progress
Complete Parallelization Experiments	4/29	Not Started
Complete Final Deliverables	5/4	Not Started

Below is another schedule representing finer-grained goals to complete the remainder of the project.

3 Work Completed So Far

So far, we have analyzed the `scikit-learn` code to determine how the random forest model is implemented and how it is trained. We have translated that procedure over to C++ using the `Eigen` matrix library instead of Python's `numpy` due to the `C++` environment that we are currently working in. We have our model training effectively and with comparable performance to the scikit-learn implementation, which gives us proof that our model is correctly implemented and that we can now parallelize and start implementing our parallel version.

4 Progress w.r.t. Goals

Compared to the goals that we had set for ourselves, we are relatively on track to complete what we had set out to complete at the start. During our proposal, we had planned to finish writing a sequential implementation and also profiling the sequential implementation by the time of the midway report. At this point, we have a working sequential implementation and have started profiling this code. From here, we will have to finish profiling and then begin optimizing the sequential code to create a parallel implementation that is more efficient.

New List of Goals:

- Implement a sequential version of Decision Trees and Random Forest in C++
- Profile this code to find inefficiencies that can be improved by parallelizing this code
- Use OpenMP to parallelize code to train trees in parallel
- Perform experiments to compare performance of sequential and parallel versions of code

5 Poster Session Plan

We plan to demo our algorithm at the poster session where we will train our Random Forest algorithm on various datasets using our sequential implementation and then show the performance improvements using our parallel version of the algorithm. We will show the comparable accuracies of the sequential and parallel algorithms which should show similar performance; however, we will show plots displaying how the parallel algorithm has strong speedup. Furthermore, we will generate plots on how speedup for training and inference time change with the number of processors for our workloads and show that our algorithms significantly improve the speedup.

6 Preliminary Results

Preliminary results include running the sequential decision tree classifier and having a working machine learning algorithm implemented in C++. As profiling is occurring right now, we are currently gathering performance results for it to determine the most important portions of the training and inference process to parallelize.

7 Concerns and other Remarks

No concerns at the moment.

8 Schedule

- **DONE:** Week of 4/2-4/8: Start on implementation of Random Forest in C++. Reference source code from `sklearn`.
- **DONE:** Week of 4/9-4/15: Finish writing implementation of Random Forest. Start on profiling of the sequential implementation. Perform experiments to show what parts of the code are the slowest and where there is the most room for improvement with parallelism.
- **In Progress:** By Midway 4/19: Finish sequential implementation and profiling. (Deep)
  - **Done:** By 4/17: finish writing the sequential implementation and finding dataset for performing profiling (Deep)
  - **In Progress:** By 4/19: complete profiling of the sequential profiling and determine the slow portions of training the random forest model and define specific conditions for experimentation with parallelized version.
- **Not Started:** Week of 4/16-4/22: Start optimizing sequential implementation using OpenMP. (Meher)
  - **In Progress:** By 4/20: complete an initial parallelization of the random forest training process by training trees in parallel
  - **Not Started:** By 4/22: improve the parallelization by performing fine-grained parallelized training by parallelizing a finer task in the training process that is expensive (determined by profiling the sequential implementation) and by parallelizing the prediction of the random forest.
- **Not Started:** Week of 4/23-4/29: Perform further experiments to compare the new parallel implementation to the original sequential implementation. (Deep)
  - **Not Started:** By 4/26: Complete experiments comparing the speedups of training the random forest model's sequential implementation versus the parallel implementation on various dataset, and then, compare the results using different parallelized components.
  - **Not Started:** By 4/29: Accumulate the results into a writeup and compare the results by running the experiments with a higher core count on the PSC machines.
- **Not Started:** Week of 4/30-5/4: Put together final deliverables and prepare for final demo. (Meher)

- **Not Started:** Begin generating plots describing the speedup of the parallel version over the sequential version for both training and prediction of the random forest model using various datasets. Furthermore, find demo that can be run during the poster session that describes the performance improvements and trade-offs considered in training the parallel random forest in comparison to the sequential one.
- **Not Started:** Produce final poster and confirm the demo that will be run during the poster session.