Project Checkpoint

SCHEDULE UPDATES:

Week 1	Sequential code (combining chess code + MCTS + minimax, implementing and getting minimax-rollout hybrid to work)
Week 2	Sequential (implementing and getting minimax-selection hybrid to work): FEN (Peter), MCTS (Sai)
Week 3	Sequential (implementing and getting minimax-selection hybrid to work) (Sai)
Week 3.5	CUDA leaf parallelism (minimax-rollout) (Peter) + Sequential (root parallelism minimax) (Sai)
Week 4	CUDA leaf parallelism (minimax-rollout) (Peter) + OpenMP root parallelism (minimax-selection) (Sai)
Week 4.5	CUDA leaf parallelism (minimax-rollout) (Peter) + OpenMP root parallelism (minimax-selection) (Sai)
Week 5	OpenMP root parallelism (minimax-selection) (Sai) + Tree parallelism (CUDA/OpenMP) (minimax-selection) (Peter)
Week 5.5	OpenMP root parallelism (minimax-selection) (Sai) + Tree parallelism (CUDA/OpenMP) (minimax-selection) (Peter)
Week 6	Tree parallelism (CUDA/OpenMP) (minimax-selection) (Peter + Sai)
Week 6.5	Report and finalization (Peter + Sai)

We ran into some delays in getting the sequential version of our code up and running. We wrote code to import the FEN into the program and a few bugs in the loading affected the MCTS running. The issue was that certain illegal moves were being considered legal since the pieces were not set to indicate these moves were illegal (castling and moving a pawn forward 2 squares) (see https://github.com/ysaibhargav/simple-chess/issues/2). This was fixed, but the MCTS still needs the minimax to be implemented so that the hybrid sequential code base for our CUDA and OpenMP implementations can take hold. The MCTS can currently solve mate-in-1 problems. Getting it to solve mate-in-n for n > 1 will require some fine-tuning of MCTS

parameters, especially the exploration coefficient. Once we are confident that the MCTS works for mate-in-{2, 3}, we will begin implementation of the hybrids.
We should be able to still provide our main goals of CUDA and OpenMP parallelism by the deadline, but the Tree parallelism is even more of a stretch goal than before. At this point, the tasks ahead of us are made up of coding and doing the work: we have a baseline c++ chess solver that we are converting to use MCTS and minimax, and we have a general idea of where we can extract parallelism for both CUDA and OpenMP setups. What we will show at the poster session will most likely be a series of graphs to show the speedup CUDA and OpenMP will provide on our chess solver setup.

Deliverables: Sequential MCTS-minimax hybrid, CUDA leaf parallelism, OpenMP root parallelism, and a comparison of the times it takes to execute each setup.
“Nice-to-haves”: Tree parallelism with CUDA or OpenMP