A simple load-balancing protocol for min-max depth-first search

James A. Riechel*

December 16, 2007

Abstract

We load-balance min-max depth-first search in a simulated parallel and distributed environment using a simple broadcast and share-half-my-work protocol. We choose a rich sub-domain of chess to search in, where en passant capture is not allowed, where neither king can castle king-side or queen-side, where the kings can be captured but are too valuable to lose, and where pawns do not and cannot promote. Four experiments are performed on two different platforms. The four experiments on both platforms use a different number of nodes or processes. These nodes or processes cooperate together to accomplish a common search task. In order, 1 node or process is used, 10 nodes or processes are used, 100 nodes or processes are used, and 1,000 nodes or processes are used, on both platforms. When an experiment is complete, each node or process reports the virtual time of its execution in virtual seconds. Each evaluation of a chess position counts as one virtual second ($1s$). A simple wood-count evaluation function is used. Time series data, the virtual time each node or process takes in each experiment on both platforms, is reported and analyzed to determine how well the simple broadcast and share-half-my-work protocol balances the load across all nodes or processes in each experiment on both platforms. Efficiency, and other metrics of performance, are computed and presented.

*jamesriechel@gmail.com