Lab Course: Efficient Programming of Multicore Processors and Supercomputers

Group 1: Jonas Mayer, Paul Preißner, Konrad Pröll

July 4, 2017

6.1 Understanding the code, run sequential version

Performance

Q: **What performance do you achieve, using best optimization flags for the compiler, if you do game searches up to depth 2,3,4,5 ?**

Using gcc with -O3 optimization (tests indicated slightly higher evalrate compared to -O2):

Measurement	**Evalrate**	**Total evals**
depth 2:		
Default	581k e/s	98,912
Midgame1	481k e/s	412,533
Midgame2	558k e/s	263,972
Endgame	572k e/s	128,197
depth 3:		
Default	589k e/s	5,045,110
Midgame1	484k e/s	34,327,884
Midgame2	561k e/s	16,256,864
Endgame	536k e/s	9,997,545
depth 4:		
Default	574k e/s	283,320,928
see note below		
depth 5:		
Default	583k e/s	>798,092,000
see note below		

Note: depths 4 and 5 were not particularly feasible to run with sequential minimax search (e.g. Default board, depth 4, single move resulted in 8:21 minutes runtime, depth 5 results in more than 25 minutes runtime). As complexity

increases exponentially with greater depth, so does runtime. Evalrate itself naturally remains the same.