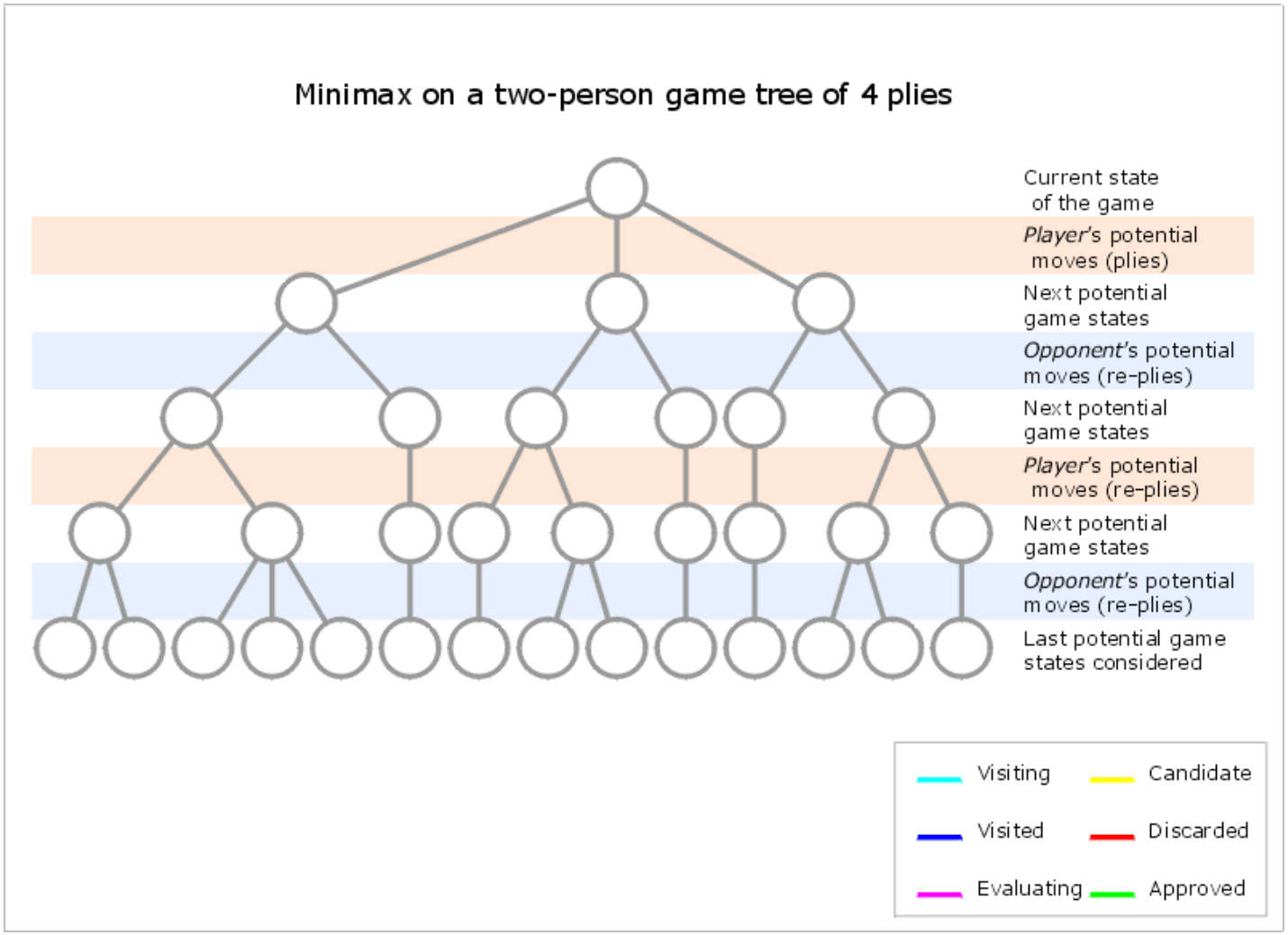
Parallelization of Game search

Shrikant Vinchurkar
Mayank Chaudhary

Important tasks

- Understanding Abalone
- Understanding the code flow
- Study of other strategies
- Developing minimax strategy

Minimax algorithm



Option -n

- changes evaluation if not set
- may produce different results every time

Best optimization flags

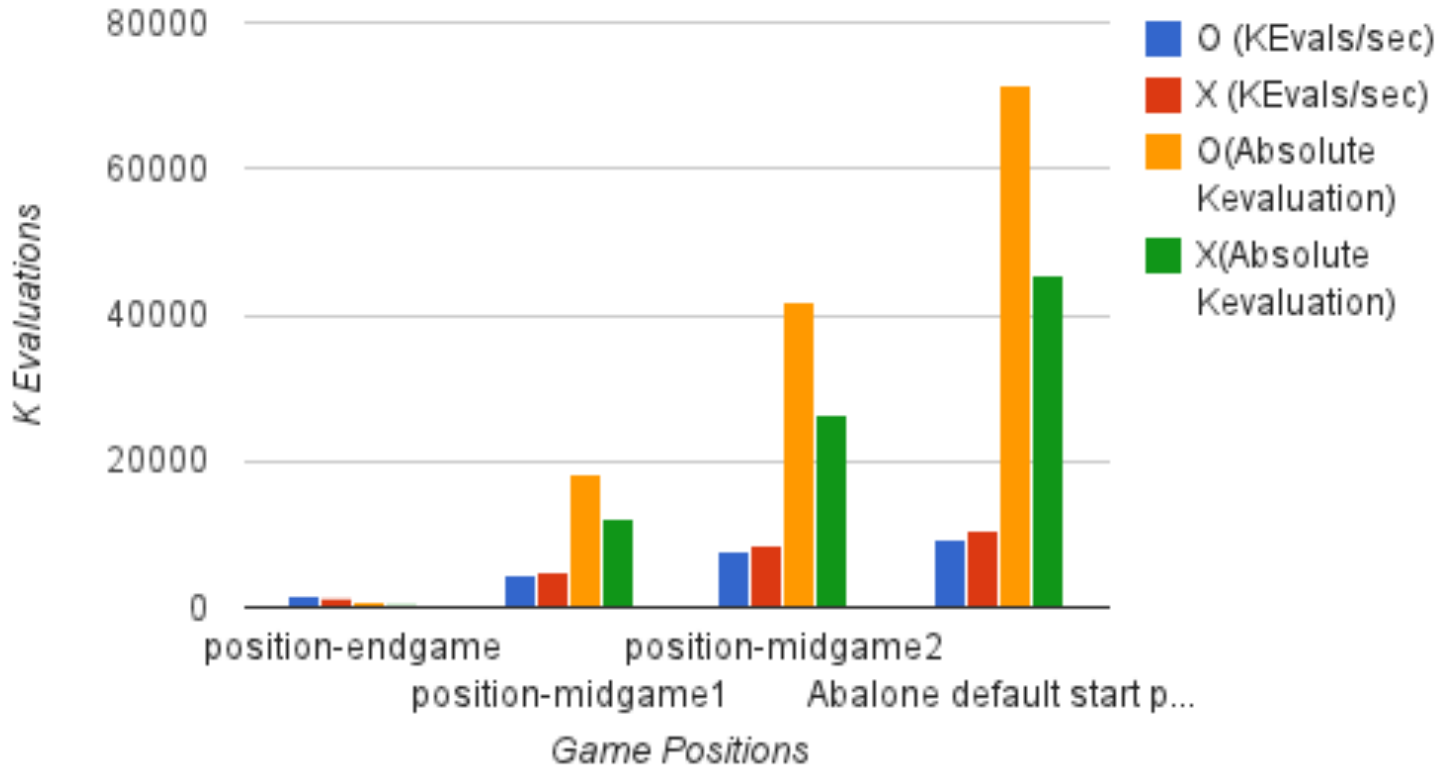
- -O3
- -fast

Evaluations/sec

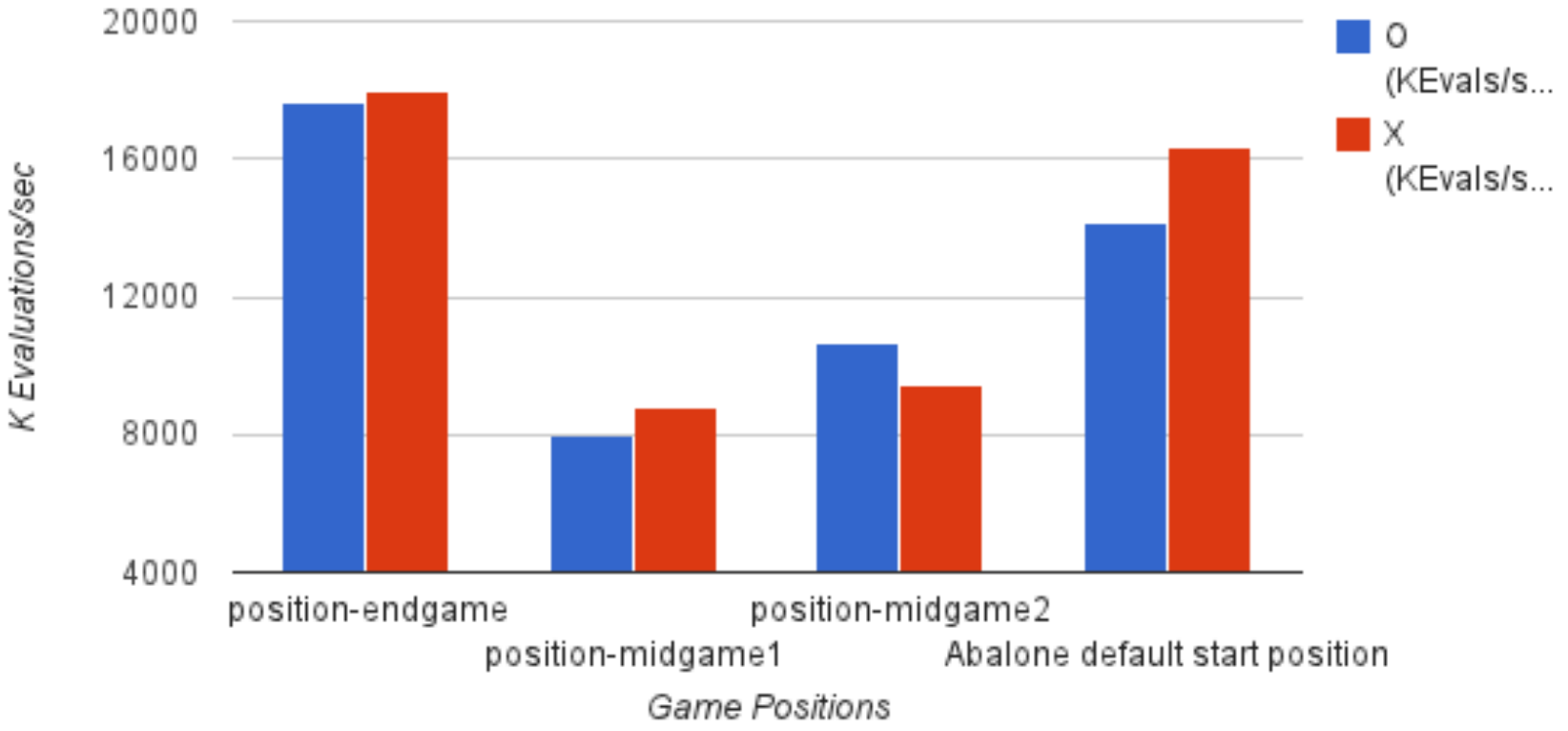
- Metric to calculate # of calls to evaluate function every sec

Measurements on Ixhalle

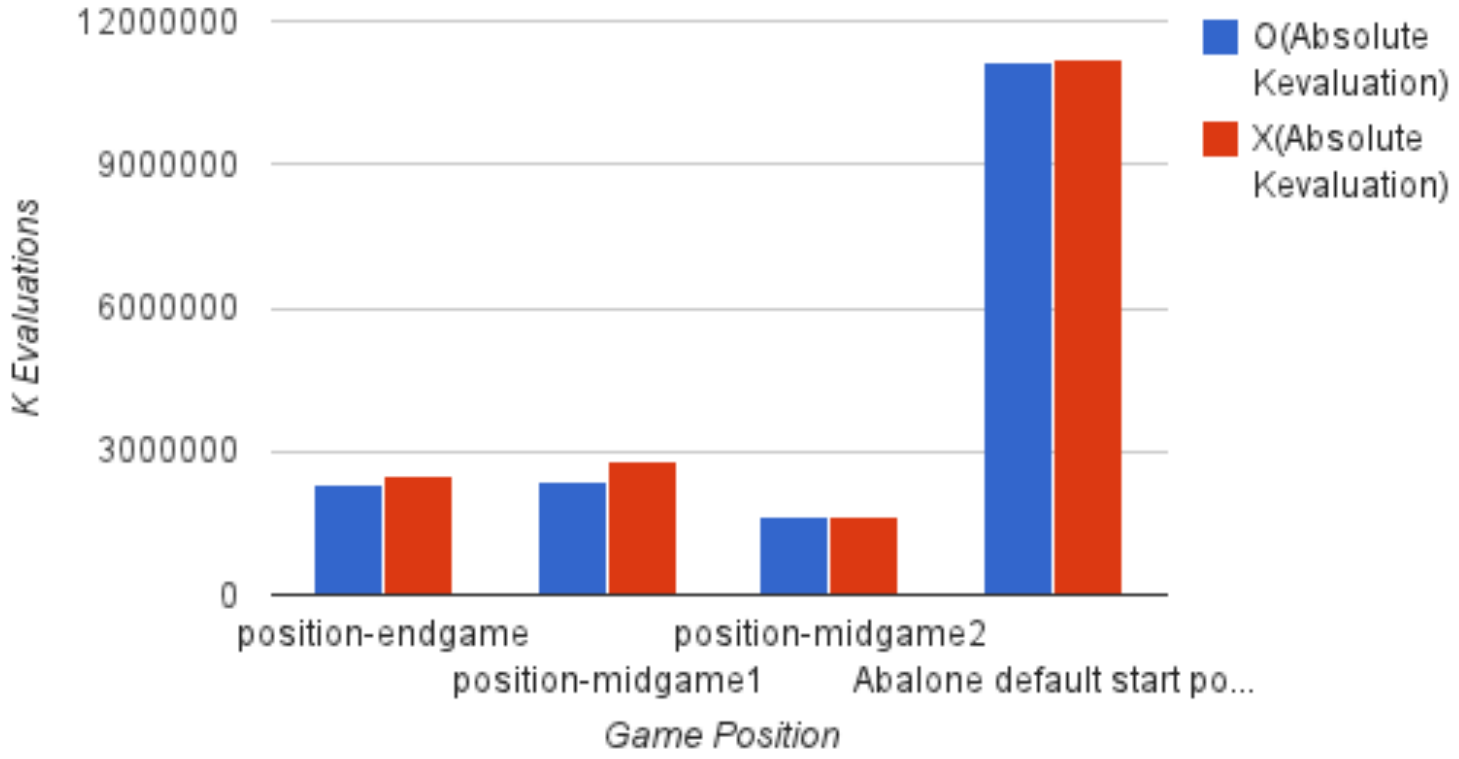
Results - Sequential, Max Depth = 2



Results- Sequential, MaxDepth-3



Results- Sequential, MaxDepth-3



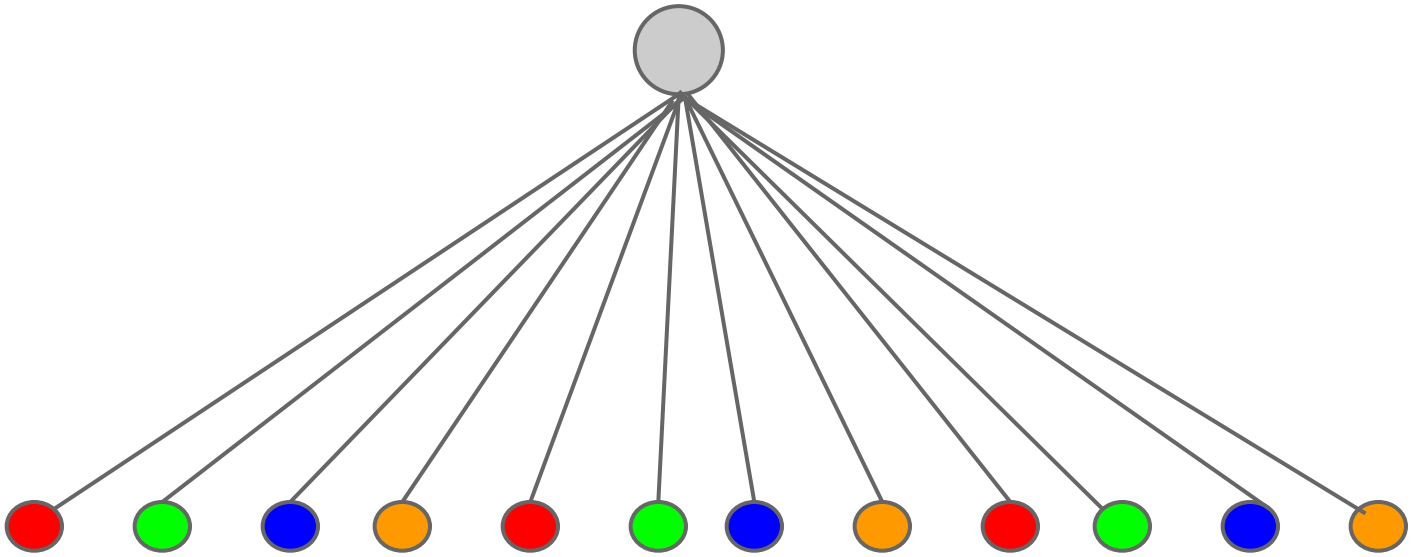
MPI parallelization of MiniMax - Approach

- Distribute work at depth 1 of minimax tree
- Assign the node evaluations in a round robin fashion - Load Balancing
- Every MPI process will roughly do equal no of node evaluations
- Every MPI process will find the best move among the moves processed by it
- Finally, every process will send its best move alongwith value to process 0
- Process 0 will determine best move of all moves

MPI parallelization of MiniMax - Approach

- Only process 0 will play actual move, change the board & broadcast it to all processes

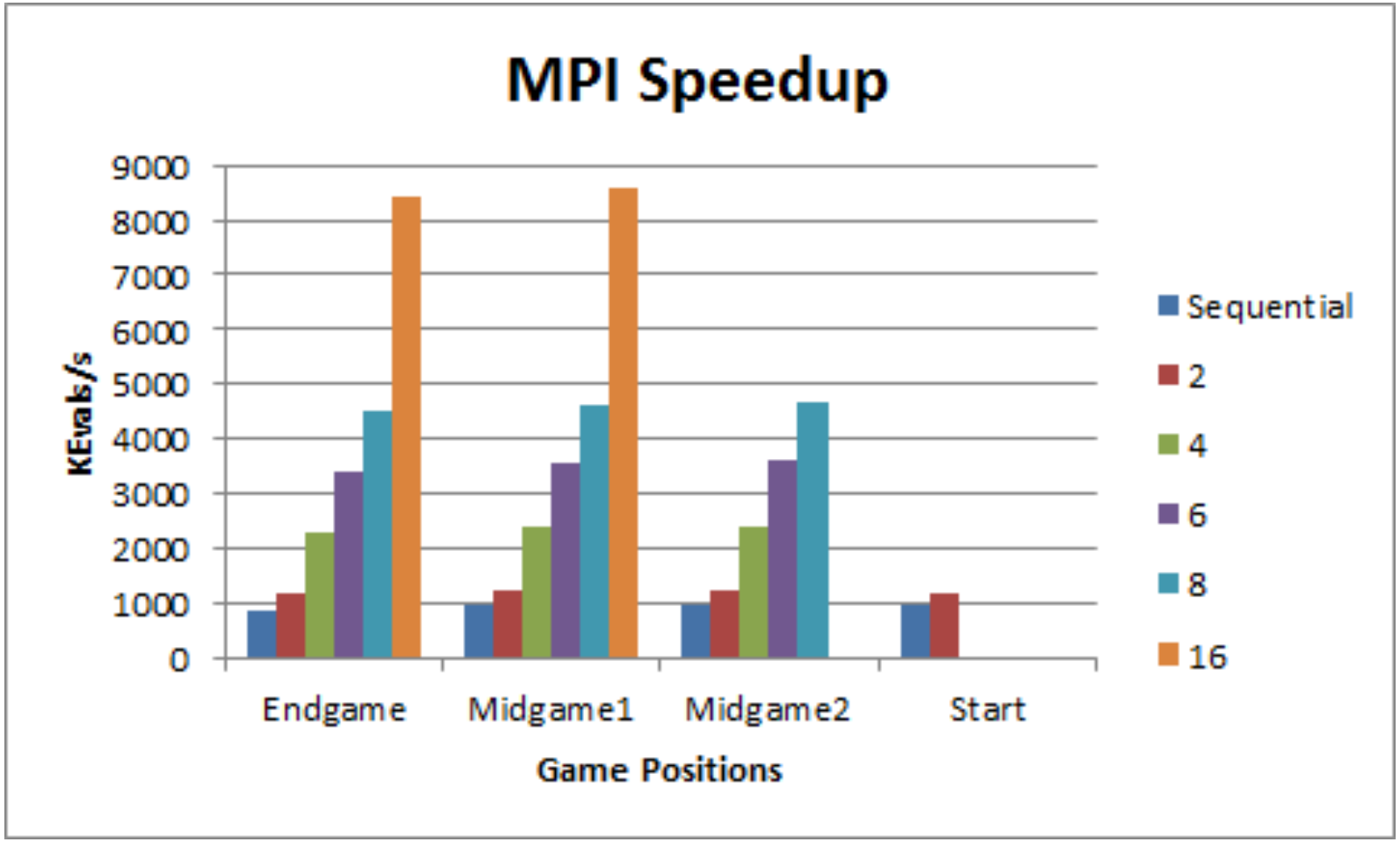
MPI parallelization approach



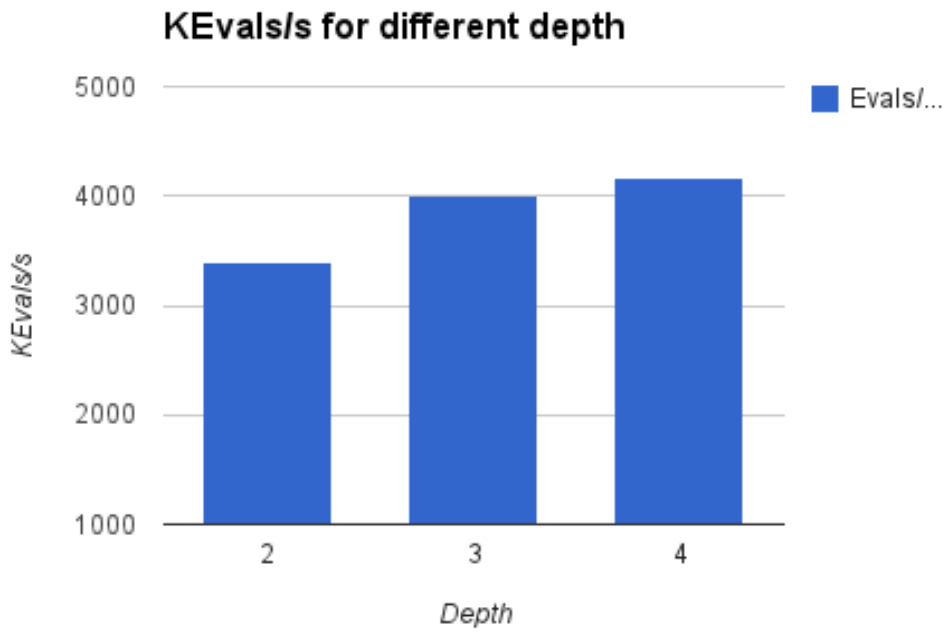
Results with MPI Parallelization

(SuperMUC)

Minimax parallelization
(Measurements for depth 2)



Measurements for different depth
(6 threads)



Depth	Time(sec)
2	0.363
4	65.186
6	833.705

Load balancing with round robin