Harvard John A. Paulson
School of Engineering
and Applied Sciences

Optimizing Data Generation For Deep Reinforcement Learning

Team 21: Rudra Barua, Charlie Harrington, Fiona Henry, Michael Krumdick

Chess and Artificial Intelligence

1948: **Alan Turing** creates the first chess “engine,” Turbochamp, an algorithm that he manually has to compute by hand.
1950: **Claude Shannon** writes *Programming A Computer For Playing Chess.*
1955: **John von Neumann** writes a chess engine for the MANIAC I.
1983: **Ken Thompson**’s chess engine, Belle, becomes the first to achieve master level play.
1997: Deep Blue beats world champion Gary Kasparov.
2017: DeepMind’s deep reinforcement learning based AlphaGoZero beats Stockfish, the best traditional chess engine.

Problem Statement

RL models are hyper optimized and efficiently process data in batches

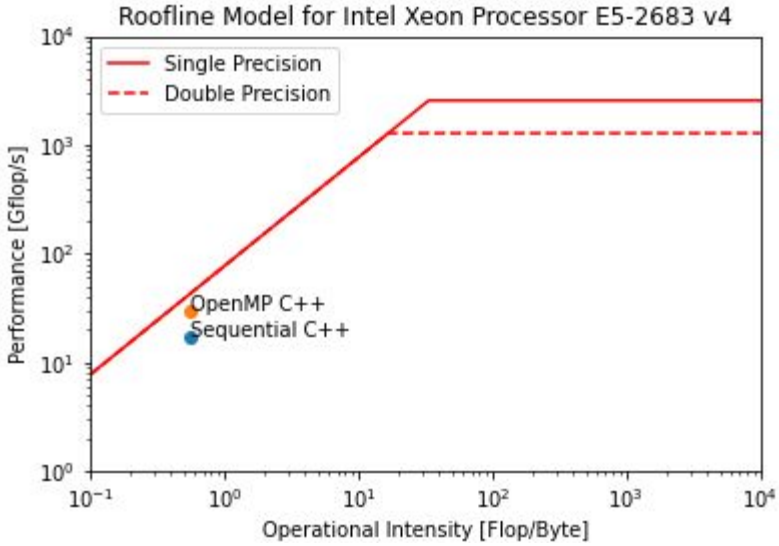
Environments models interact with are problem specific and often written in sequential Python code

Bottleneck becomes generating data in environment

Goal: Implement and parallelize chess environment generation in C++ and measure improvement in training speed

Roofline

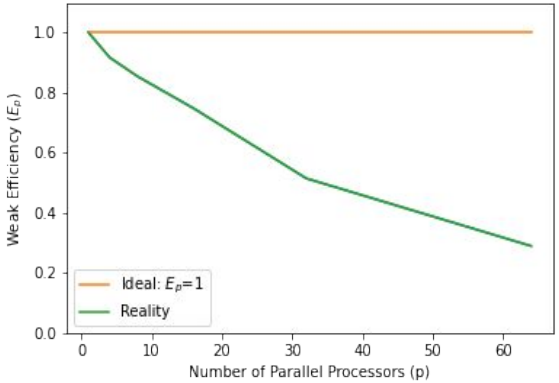
- CPU: Intel Xeon Processor E5-2683 v4
- Peak memory bandwidth: 76.8 GB/s
- Nominal peak arithmetic performance:
  - Single precision: 2560 Gflop/s
  - Double precision: 1280 Gflop/s
- Hardware Counter with PAPI (1,000,000 Boards):
  - Total instructions: 168,871,250,061
  - Instructions per cycle: 1.59331
  - Total L1 data misses: 14,006,475
  - Total load/store: 108,552,964,772
  - OI (Flops/Byte) ≈ 0.555657
  - Performance [Gflop/s]:
    - Sequential: 16.91084
    - OpenMP: 29.7934

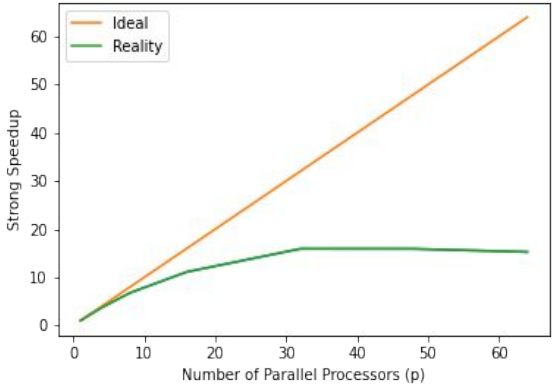


Scaling Analysis

Diminishing returns from adding more threads came earlier than expected

Decided to look more into model specific performance





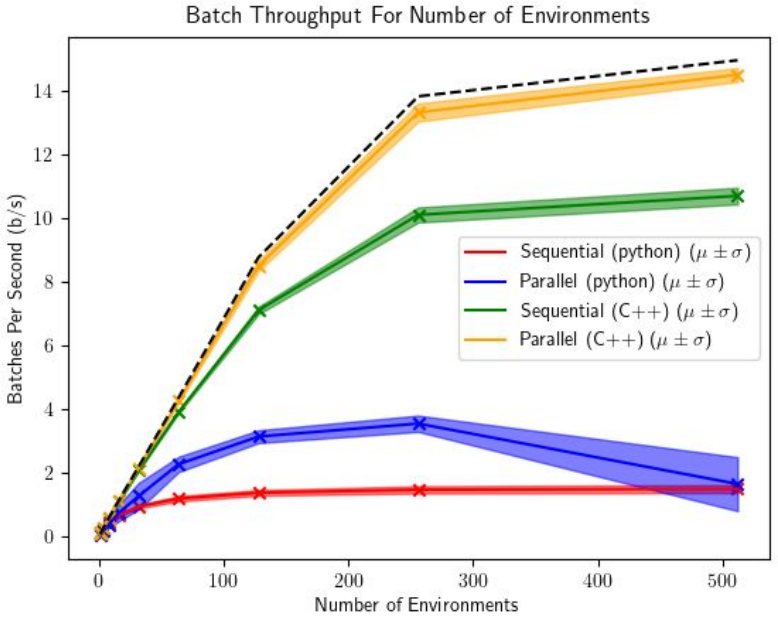
Initial Performance Results

Incorporate environment-agent interaction

Close to hitting theoretical max with shared memory parallelism

Now bound by model speed again

Time to look for more ways to optimize



chessenv: A High Performance RL Environment for Chess

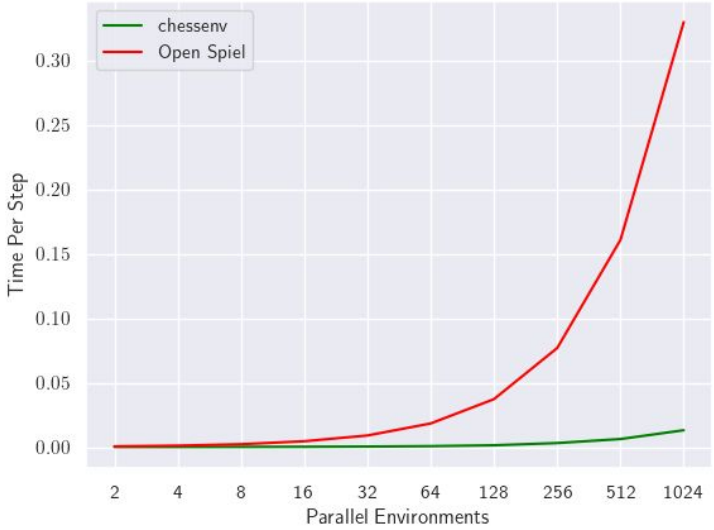
Fastest open source chess environment for reinforcement learning

Used a bitboard move generation library in C

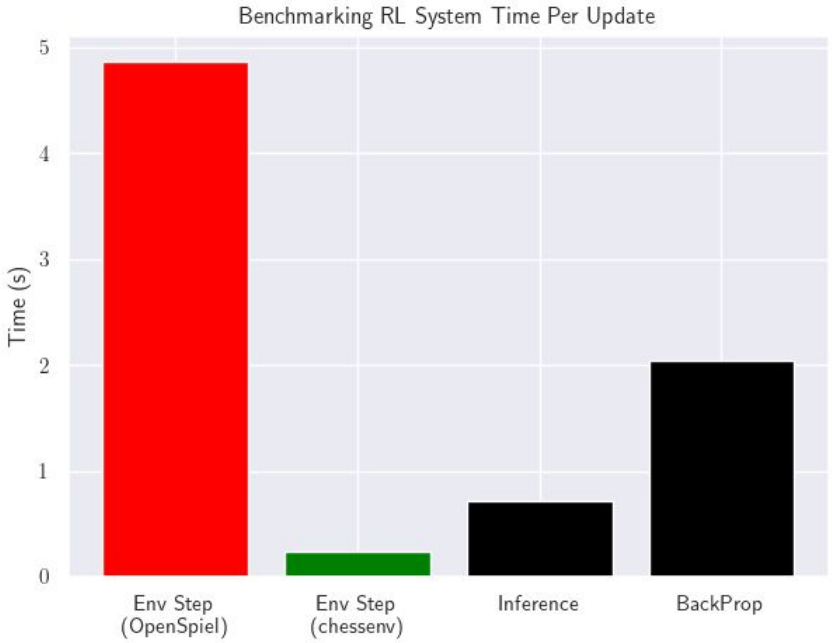
Parallelized using openmp and shared memory

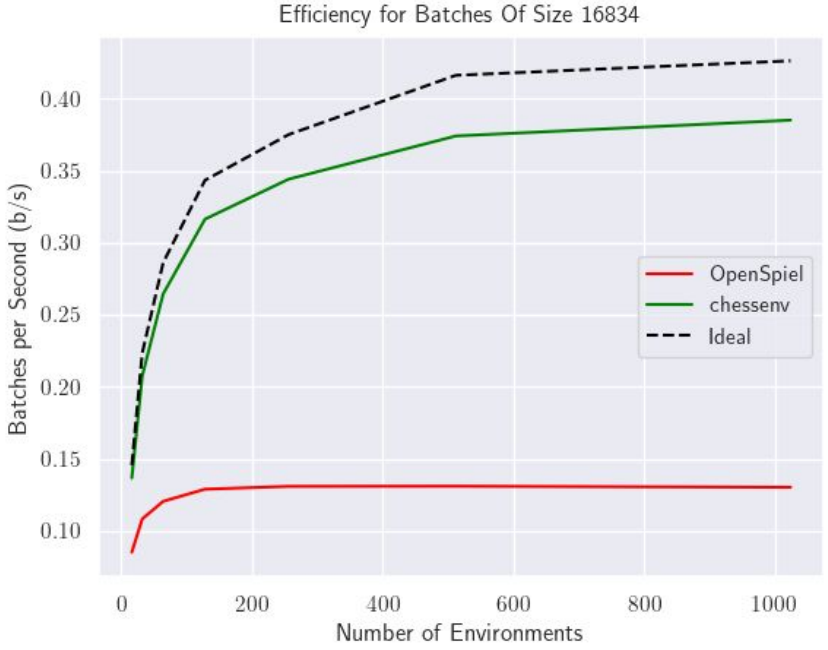
Used CFFI and numpy to efficiently send data from C arrays to pytorch tensors

Benchmarked performance against DeepMind's OpenSpiel



chessenv vs OpenSpiel

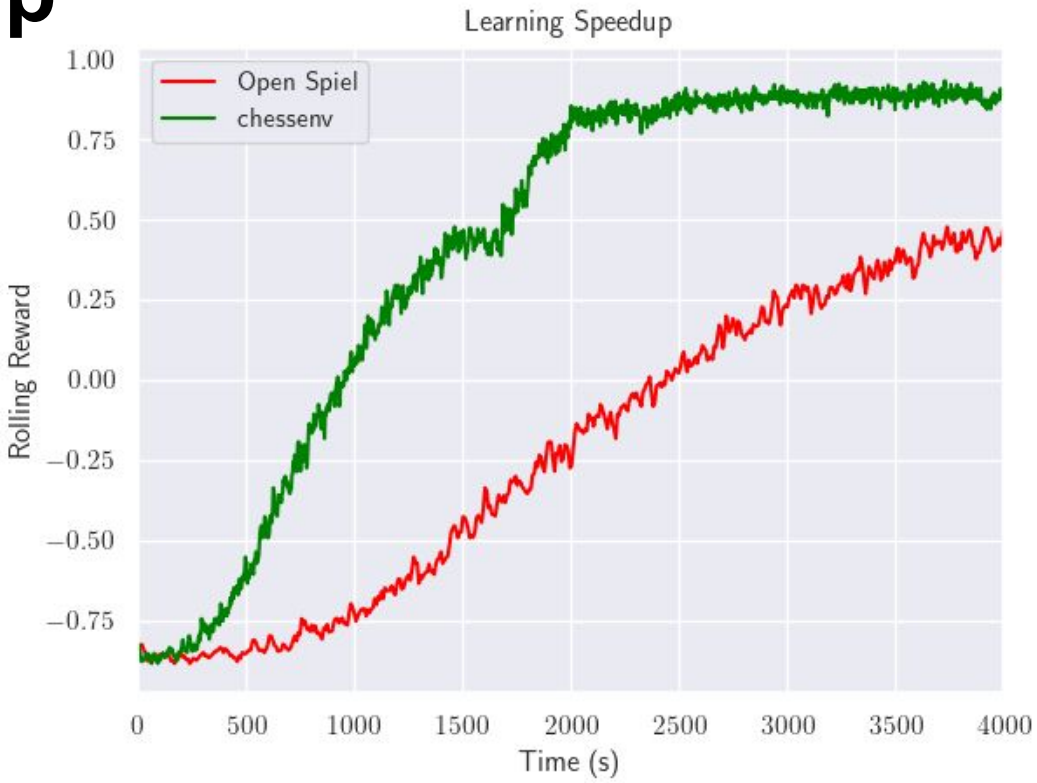




Learning Speedup

We can use our performance statistics to estimate how much faster our learning process is with our environment

Ran on google cloud instance with 4 Intel Xeon CPUs (2.30GHz) and one T4 GPU



Insights

More cores is not always the answer - look at the scaling!

Get closer to hardware before naively parallelizing

Fixing python bottlenecks in C or C++ is a powerful framework for increasing performance

Future Work

Optimizing other parts of the RL system to speed up training even more!

Apply what we learned about getting speed ups using C to other machine learning problems

Questions?

Harvard John A. Paulsor
School of Engineering
and Applied Sciences