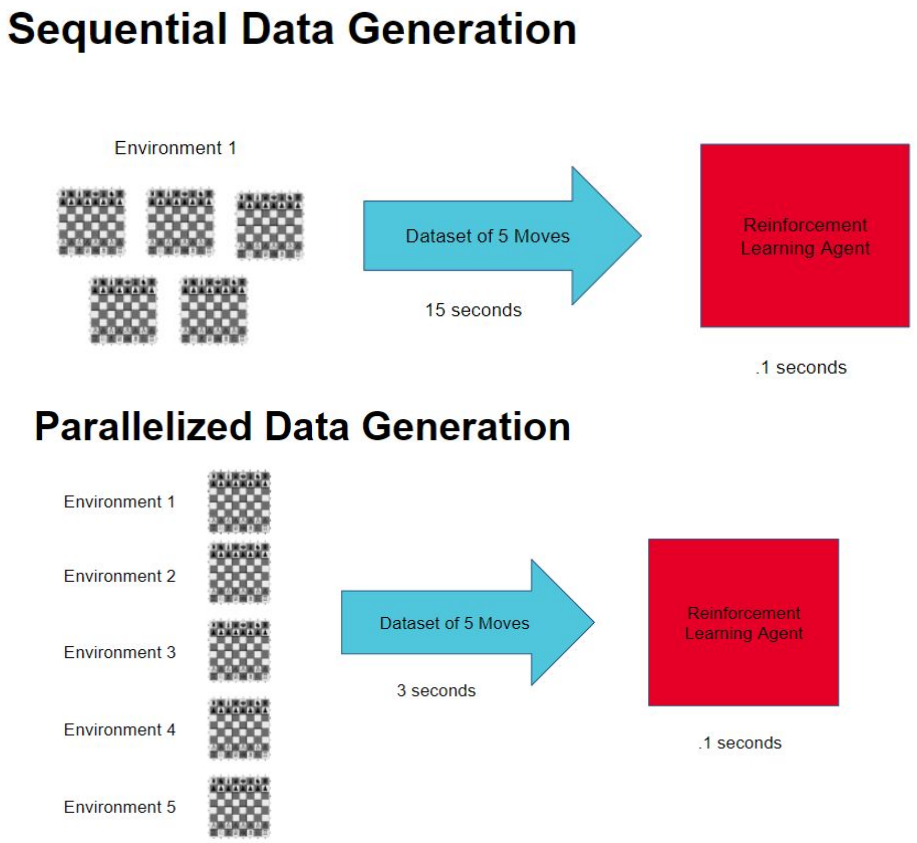
Harvard John A. Paulson
School of Engineering
and Applied Sciences

Optimizing Data Generation for Reinforcement Learning

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

Background

- **Goal:** Optimize data generation for training a deep reinforcement learning (DRL) agent to learn chess
- We have a pre-existing DRL system we can use to benchmark increases in performance
- We can compute the estimated speed up without re-running the learning algorithm

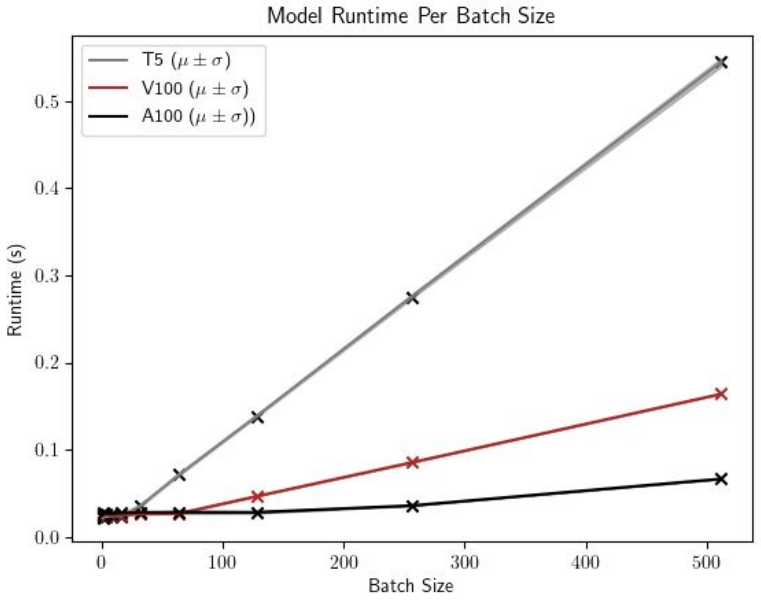


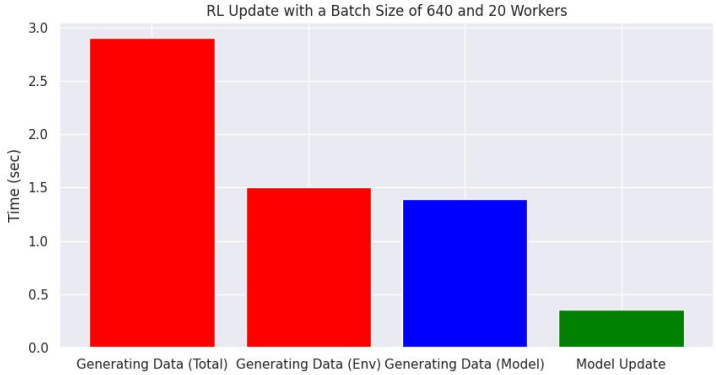
Benchmarking

An important factor in the runtime is the speed of the model itself

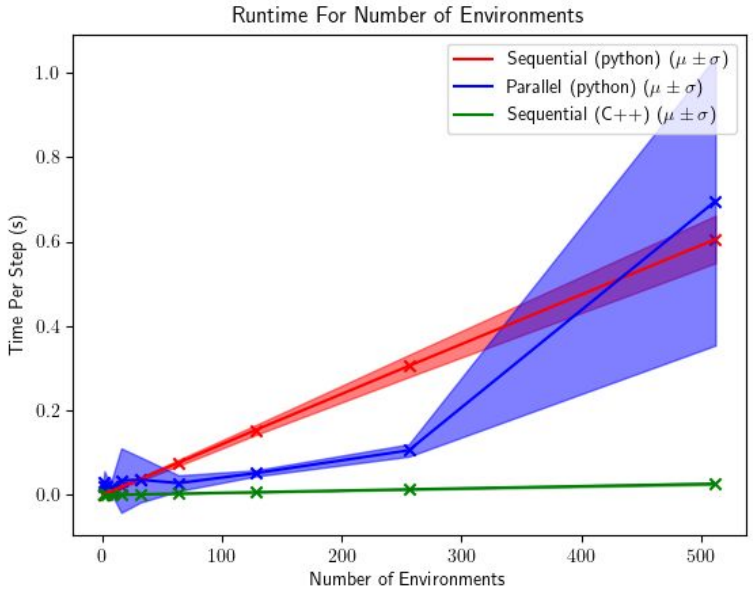
Time to generate batch of 512 =
# of steps need to reach 512 * (Time to generate data + Time to generate actions)

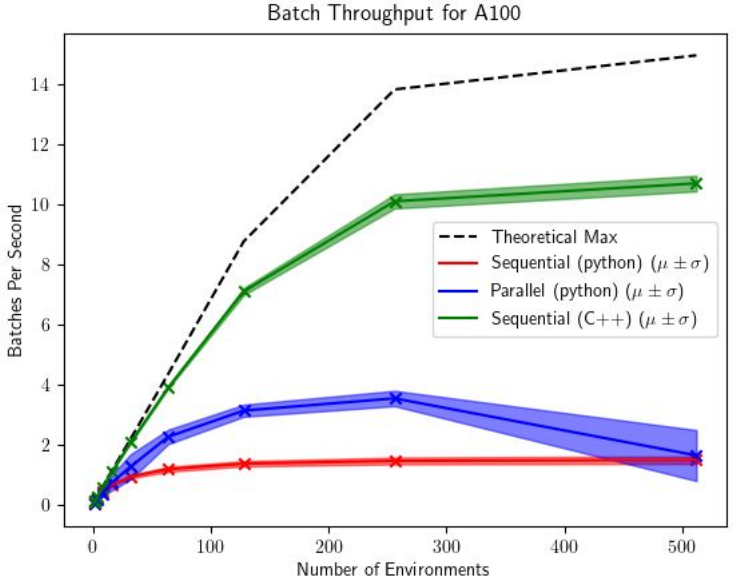
Benchmarked performance across 3 major GPUs





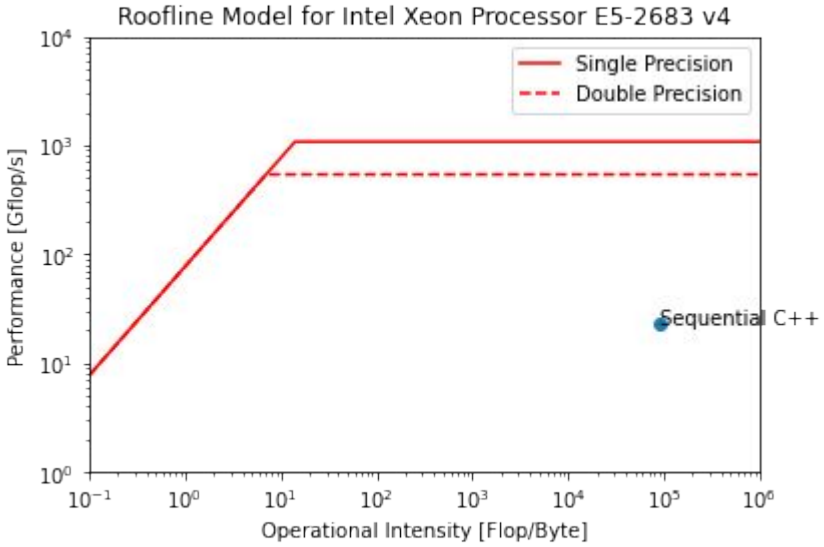
Baseline Benchmarking





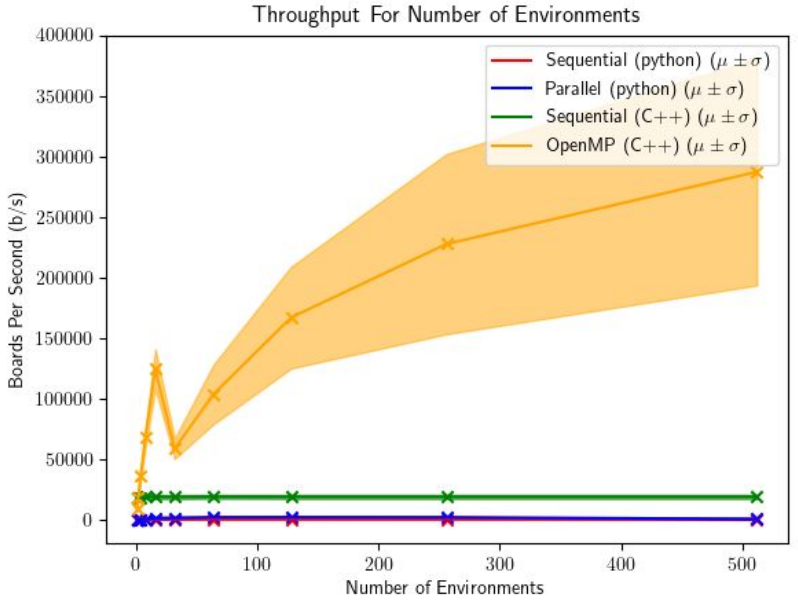
Roofline Model

- CPU: Intel Xeon Processor E5-2683 v4
- Peak memory bandwidth: 76.8 GB/s
- Nominal peak arithmetic performance:
  - Single precision: 1075.2 Gflop/s
  - Double precision: 537.6 Gflop/s
- Hardware Counter with PAPI (1000 Boards):
  - Total cycles: 105227231
  - Total instructions: 168874237
  - Instructions per cycle: 1.60485
  - Total L1 data misses: 16081
  - Total load/store: 108550822
  - OI (Flops/Byte): 9.126484e4
  - Performance [Gflop/s]: 2.25453



Basic Parallelization

- Just added a single #pragma omp parallel statement!
- Expect linear graph: graph flattens out due to resource saturation
- Some weird memory issues to work through still

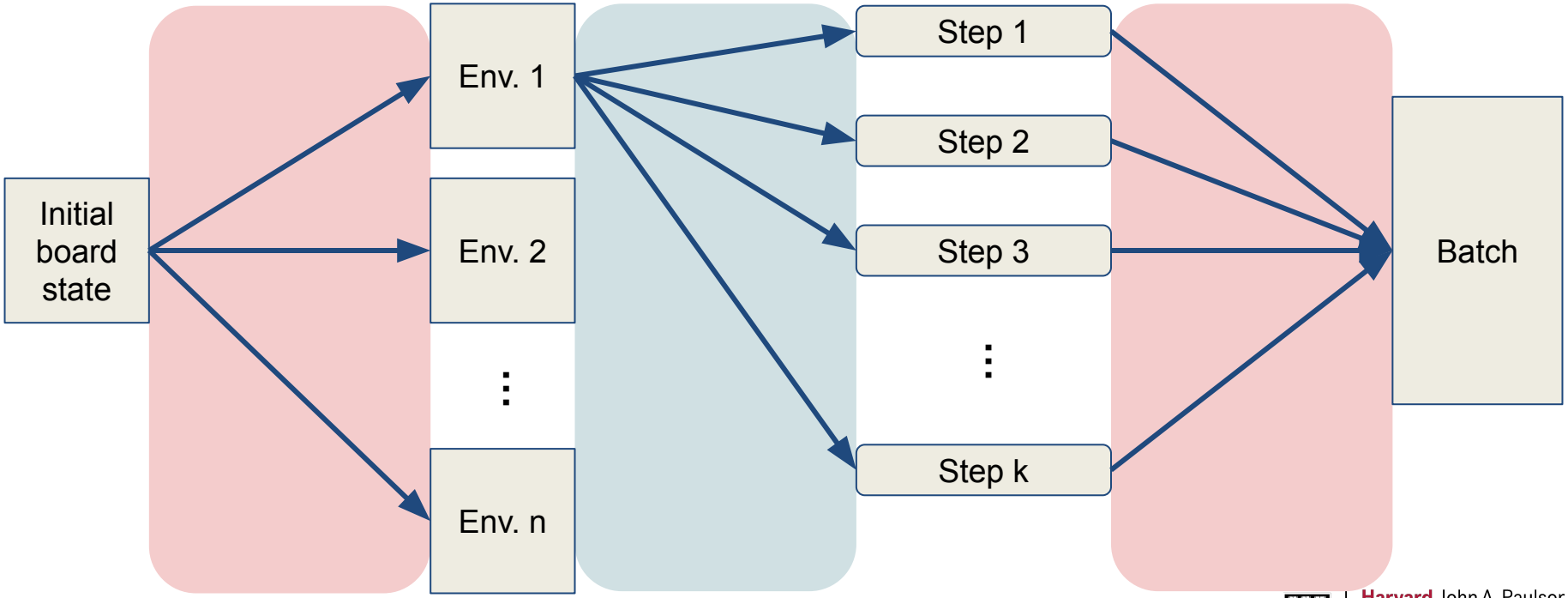


Future Parallelism

- Shared memory: parallelize single environment with OpenMP (like basic parallelism shows)
- Distributed memory: run more environments by distributing with MPI (one environment per core)
  - Analyze overhead impact (might not be worth it)

Implementation	Environment Generation Speed-up	Estimated Impact on RL Performance
Python	1.0x	-
Parallelized Python	-	-
C++	-	-
C++ with OpenMP	-	-
C++ with OpenMP & MPI	-	-

Future Parallelism



Questions?