NAVIX: Scaling MiniGrid with JAX

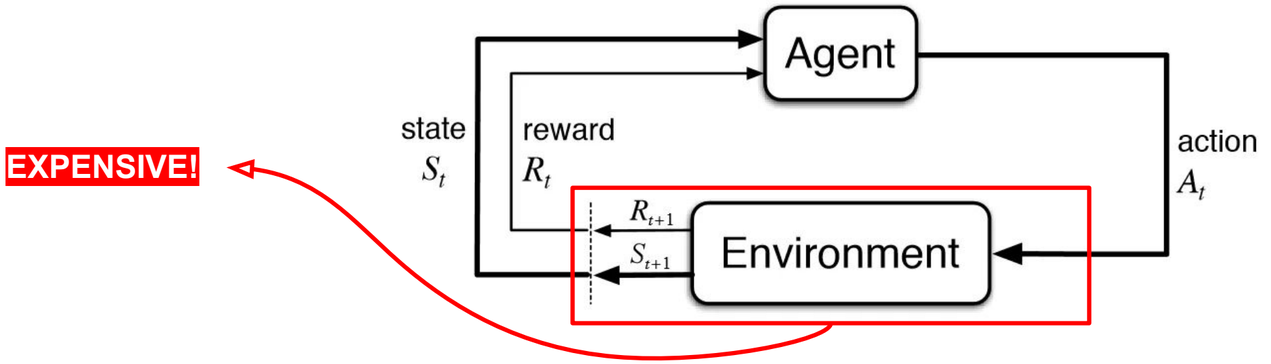
E. Pignatelli *J. Liesen* *R. T. Lange* *C. Lu* *P. S. Castro* *L. Toni*

Paper: *https://openreview.net/forum?id=lPVz01z0DI*
GitHub: *https://github.com/epignatelli/navix*
Docs: *https://epignatelli/navi*

Motivation

*Environments are a **bottleneck** for **experimentation***

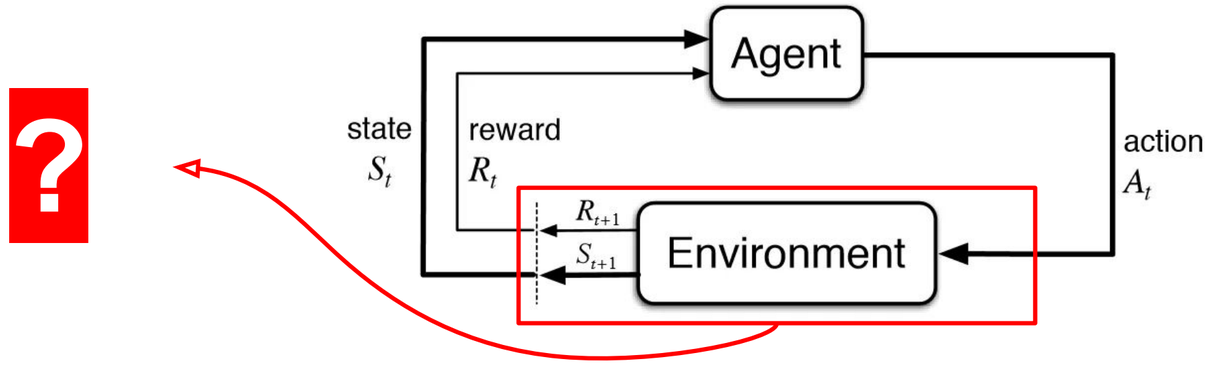
- ➔ RL experimentation is **slow** and **expensive**.
- ➔ **Why?** A great portion of the cost is in **executing environment** steps.
- ➔ Most environments implement **CPU** computations, **limiting speed** and **scalability**
- ➔ TL;DR; Environments are a **bottleneck for RL experimentation**



Potential

*Can we run **PPO** in **15 minutes**?*

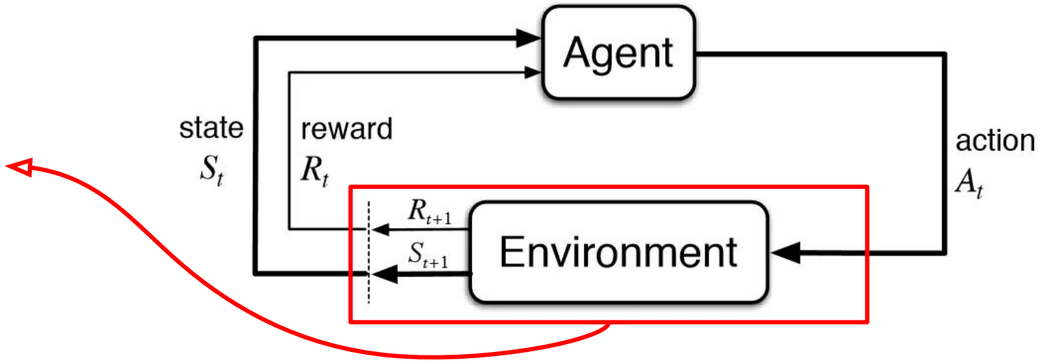
- ➔ **What if environments could scale** as well as RL algorithms do?
- ➔ Can we test a **new algorithm** in **15 minutes** on a full set of environments?
- ➔ **What environments** have the most impact?



Overview

*Reimplementing **MiniGrid** in **JAX***

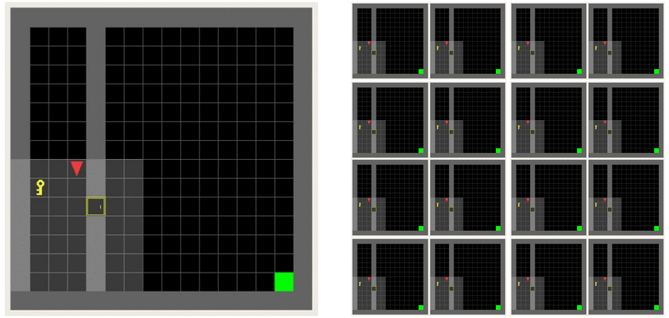
- ➔ We focus on **MiniGrid**, for its **ubiquity** as a **testbed** and relevance in RL experimentation (exploration, credit assignment, instructions, …)
- ➔ Here, we **reimplement** MiniGrid in **JAX**: **NAVIX**
- ➔ NAVIX is part of a **broader set** of **JAX reimplementations**:
  - Mujoco
  - Classic Control
  - **...**
  - *// the list is long*
- ➔ NAVIX achieves **160 000x speedup** in batch mode, compared to MiniGrid



Methods

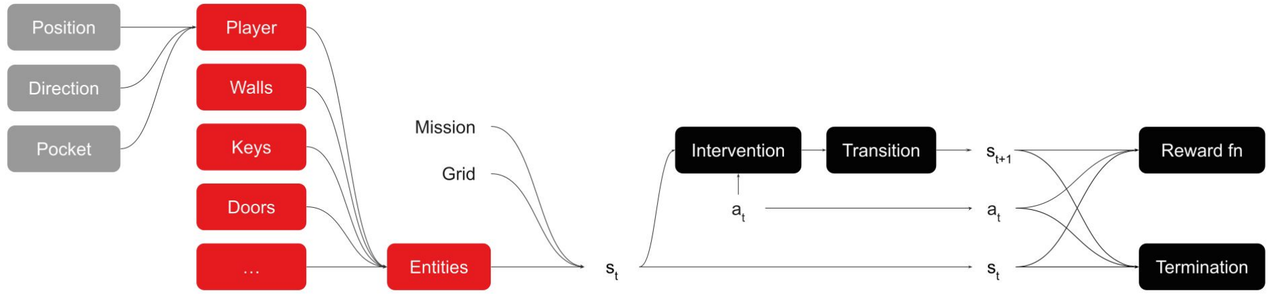
Entity-Component-Systems model + stateless computation

- ➔ NAVIX **matches** the **MiniGrid semantics** (same obs, actions, transitions, terminations, rewards, …)
- ➔ A NAVIX (Python) program can be **compiled into** an **XLA** program
- ➔ Transition from the MiniGrid **stateful** computation **to** a **stateless** one
- ➔ **Entity-Component-Systems** (ECS) model for easy **expansion**



MiniGrid *NAVIX*

Identical semantics, but faster and highly parallel.

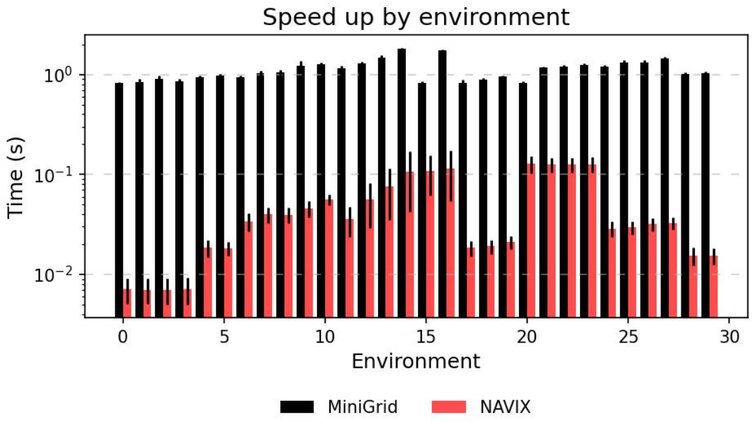


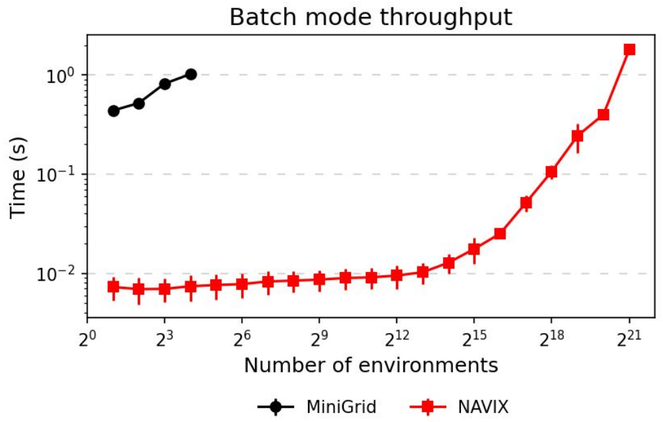
ECS model

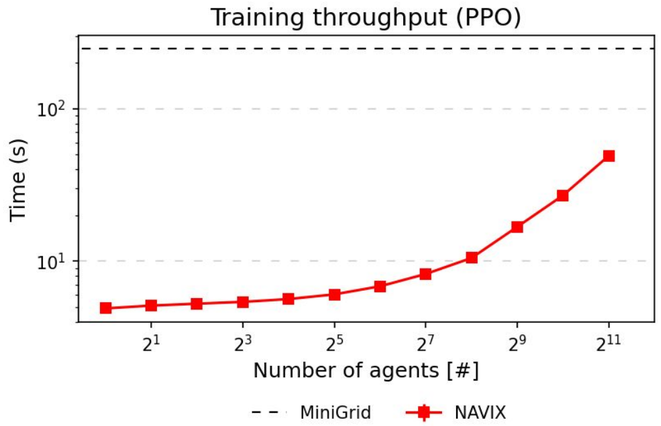
Results

Speed and Throughput - NVIDIA A100

- ➔ Is NAVIX **faster** than MiniGrid? **Yes**.
- ➔ **How much** faster? **~45X**.
- ➔ Is it more **scalable**? **Yes**.
- ➔ **How much** more scalable? **2048 PPO** agents in parallel.







Results

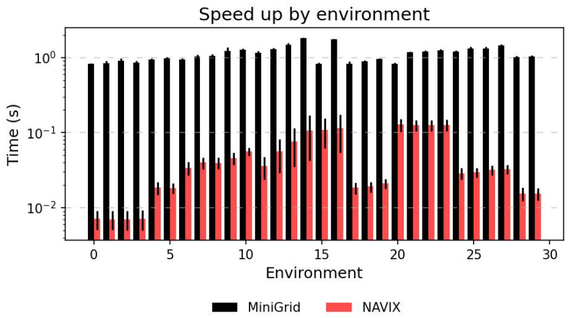
Consumer-grade hardware - NVIDIA A4000

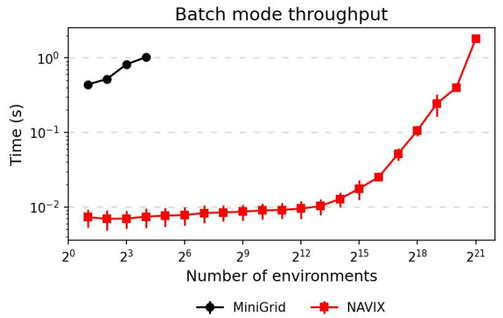
➔ Do gain transfer to **consumer-grade hardware**? **Yes**.

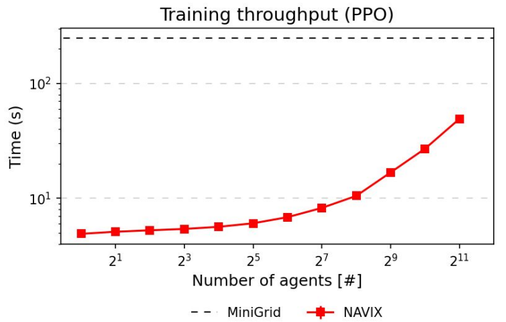
➔ **7x faster**

➔ **Similar throughput**

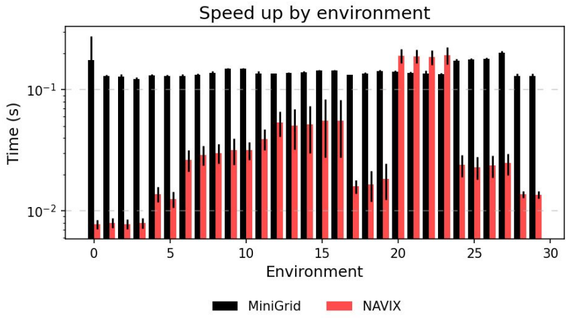
Cluster (A100 GPU)

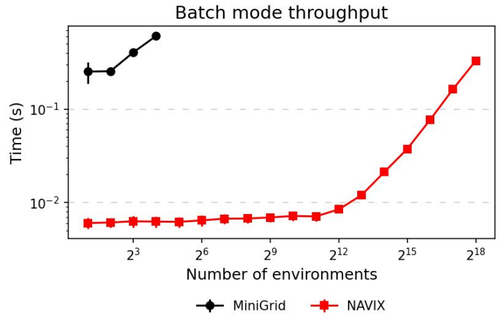


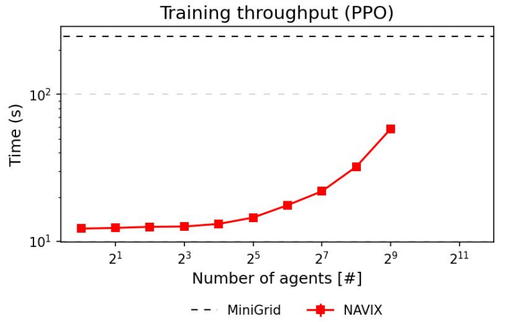




Consumer-grade (A4000 GPU)



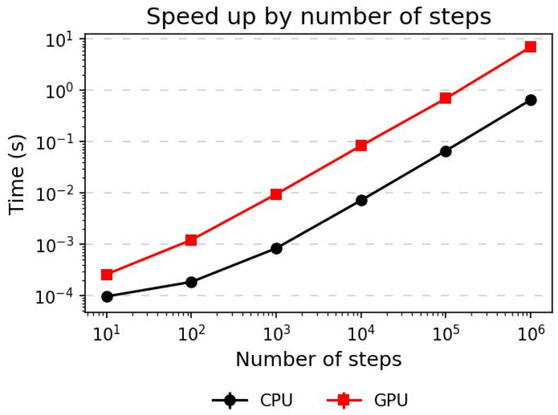


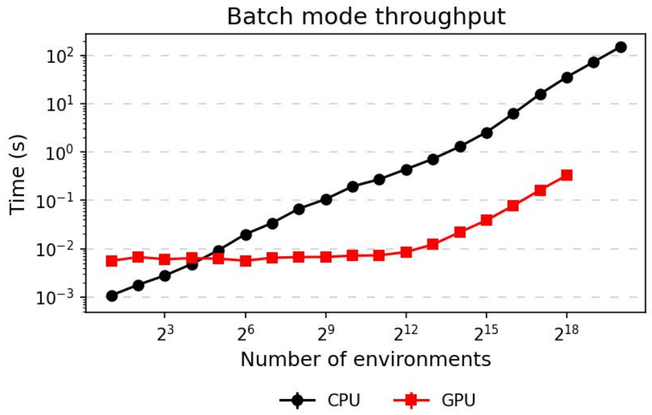


Results

Marginalising the performance over the design pattern

- ➔ **What makes NAVIX faster** than MiniGrid?
- ➔ Is the **ECS** model**?** The transition to **JAX?** Or the ability to run on **GPU?**
- ➔ We test the exact **same** NAVIX **program** on **GPU** and **CPU**
- ➔ Results show that **NAVIX** is still **faster** than MiniGrid **on CPU**.

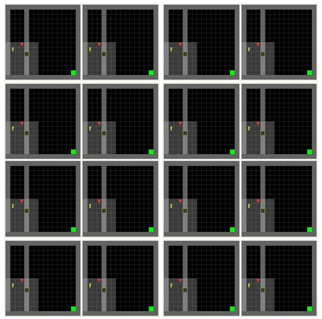


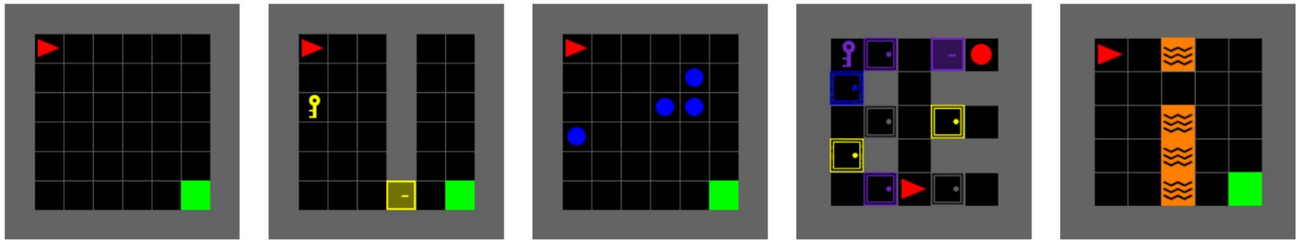


Conclusions

TL;DR

- ➔ We implemented **NAVIX**: **MiniGrid** in **JAX.**
- ➔ **Same** obs, actions, transitions, terminations, rewards, full envs of MiniGrid.
- ➔ NAVIX is over **160 000x faster** than MiniGrid in batch mode.
- ➔ Allows training more than **2048 PPO agents** in parallel.
- ➔ This turns **1-week** experiments **into 15-minutes** one**.**
- ➔ This is not just a change is speed. It's a **change of paradigm**, opening avenues unfeasible before.





(a) 128.98× (b) 26.47× (c) 41.17× (d) 19.72× (e) 45.71×

Join in!

Try it, develop it, read it, cite it!

- ➔ **Install it:** `pip install navix`
- ➔ **Docs**: *https://epignatelli.com/navix*
- ➔ **Contribute:** *https://github.com/epignatelli/navix*
- ➔ **Read the paper**: *https://openreview.net/forum?id=lPVz01z0DI*