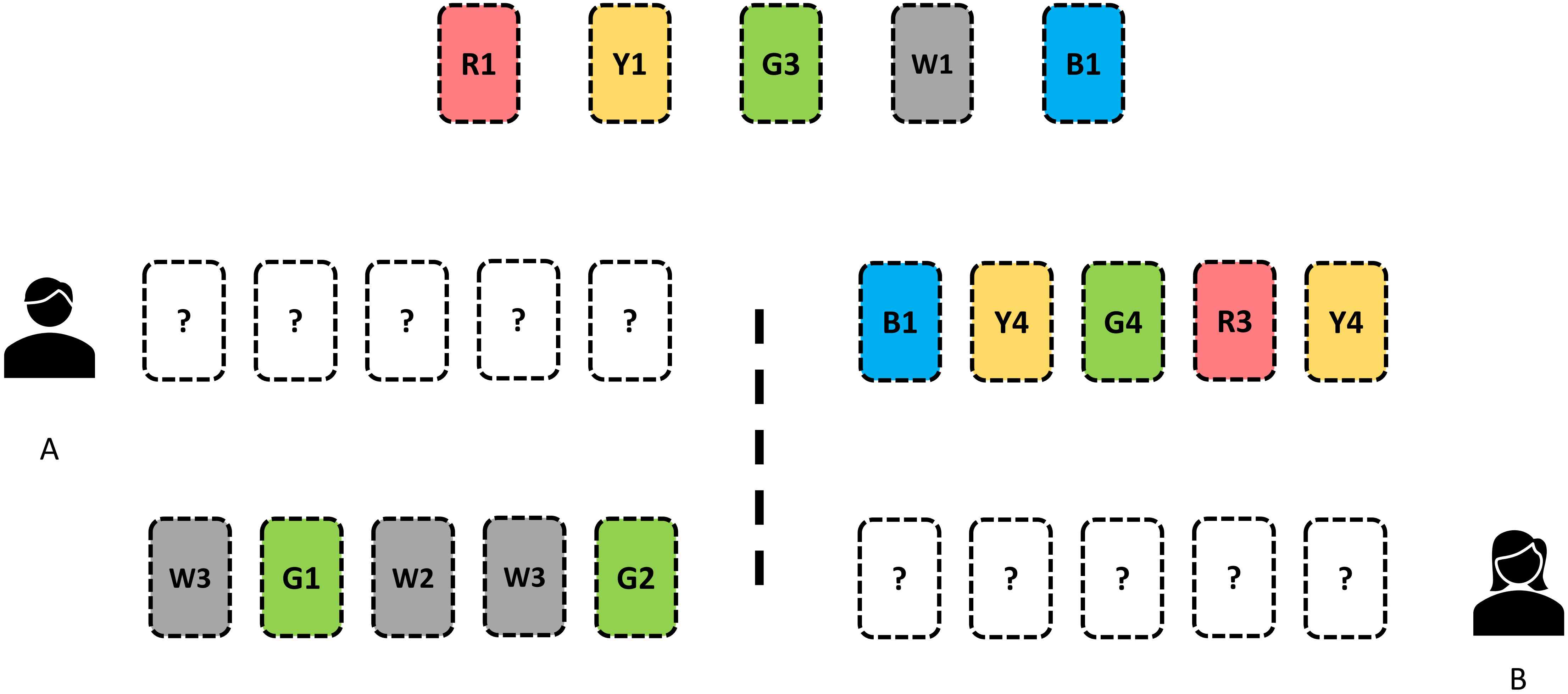
Jun Tian
https://tianjun.me/about
Software Engineer @ Microsoft, Beijing

Three Messages

- **Julia is easy to interact with other languages**
- **Julia is productive to provide HTTP services**
- **Julia is FAST and easy to use for Reinforcement Learning!**



R1 Y1 G3 W1 B1

? ? ? ? ?

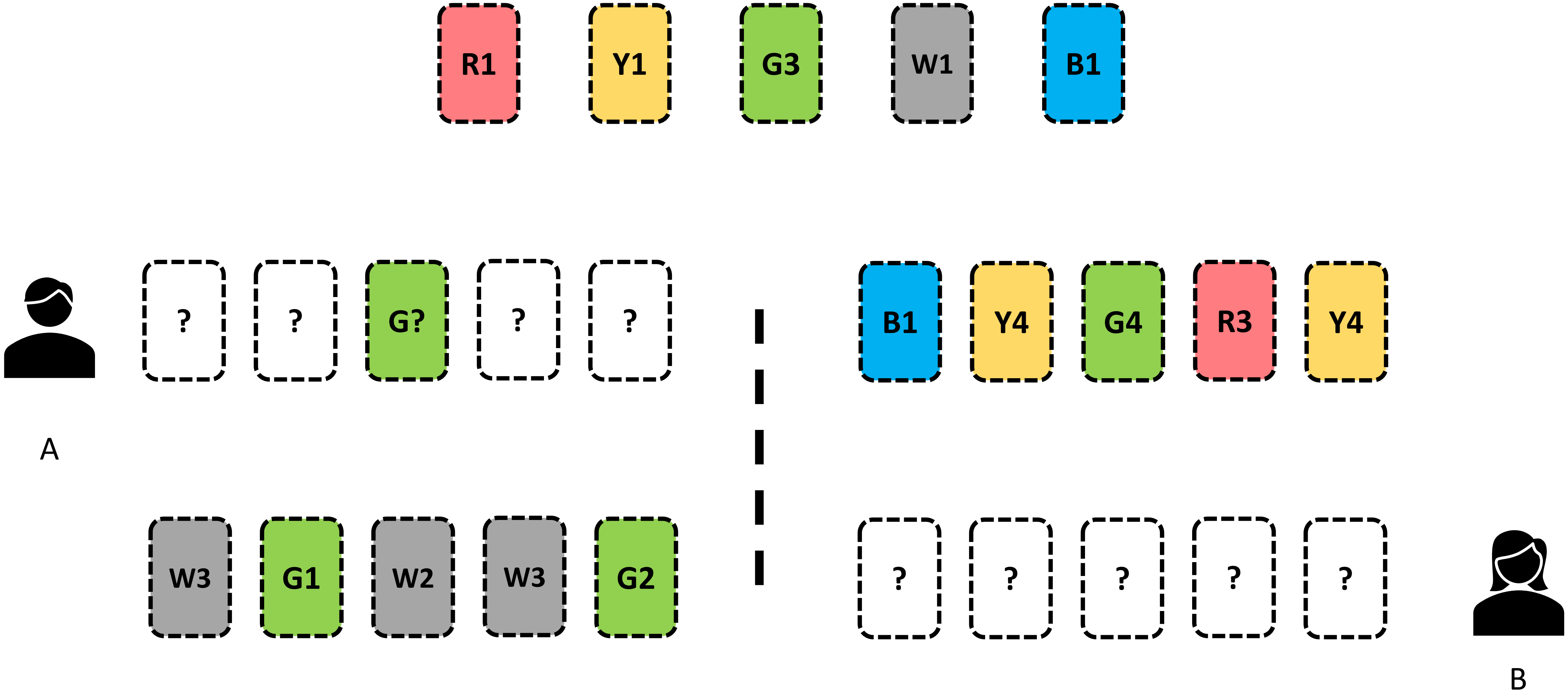
B1 Y4 G4 R3 Y4

A

W3 G1 W2 W3 G2

? ? ? ? ?

B



R1 Y1 G3 W1 B1

? ? G? ? ?

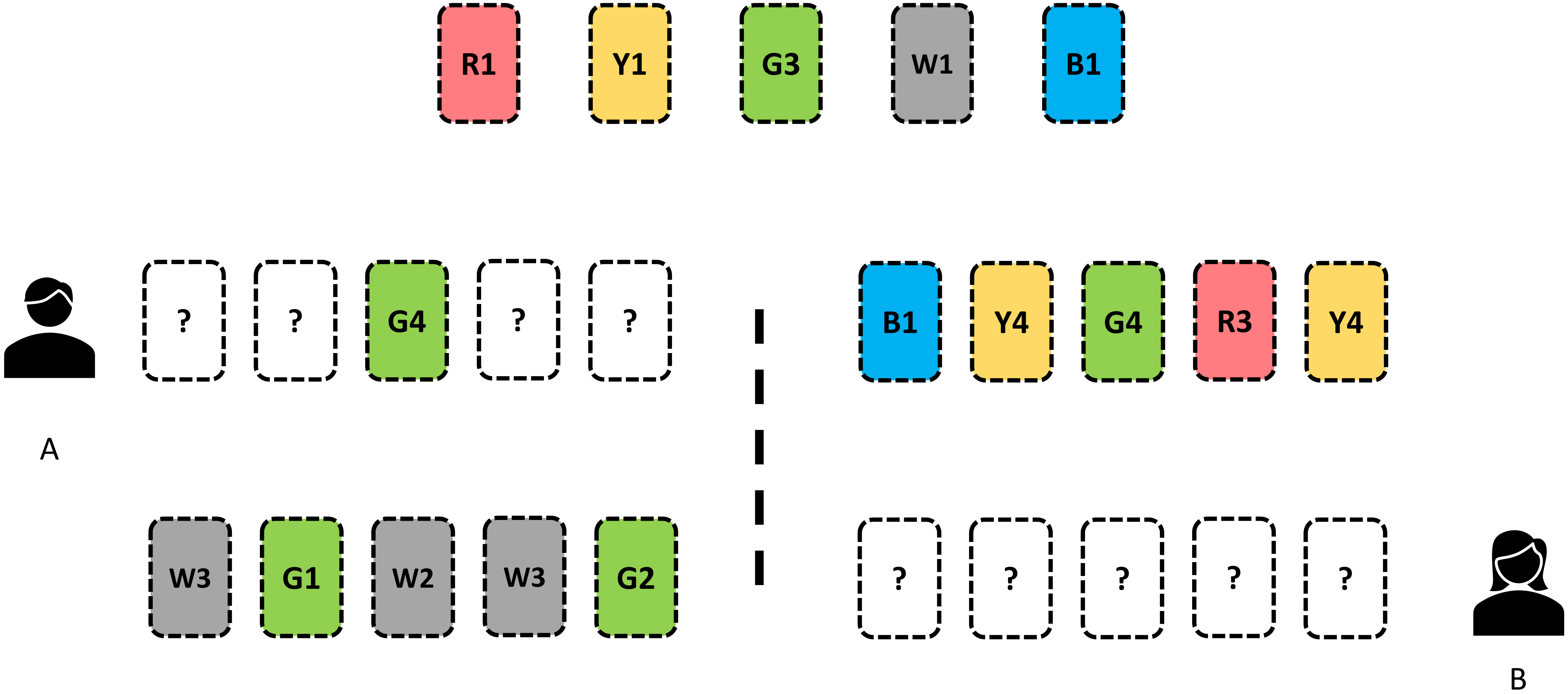
B1 Y4 G4 R3 Y4

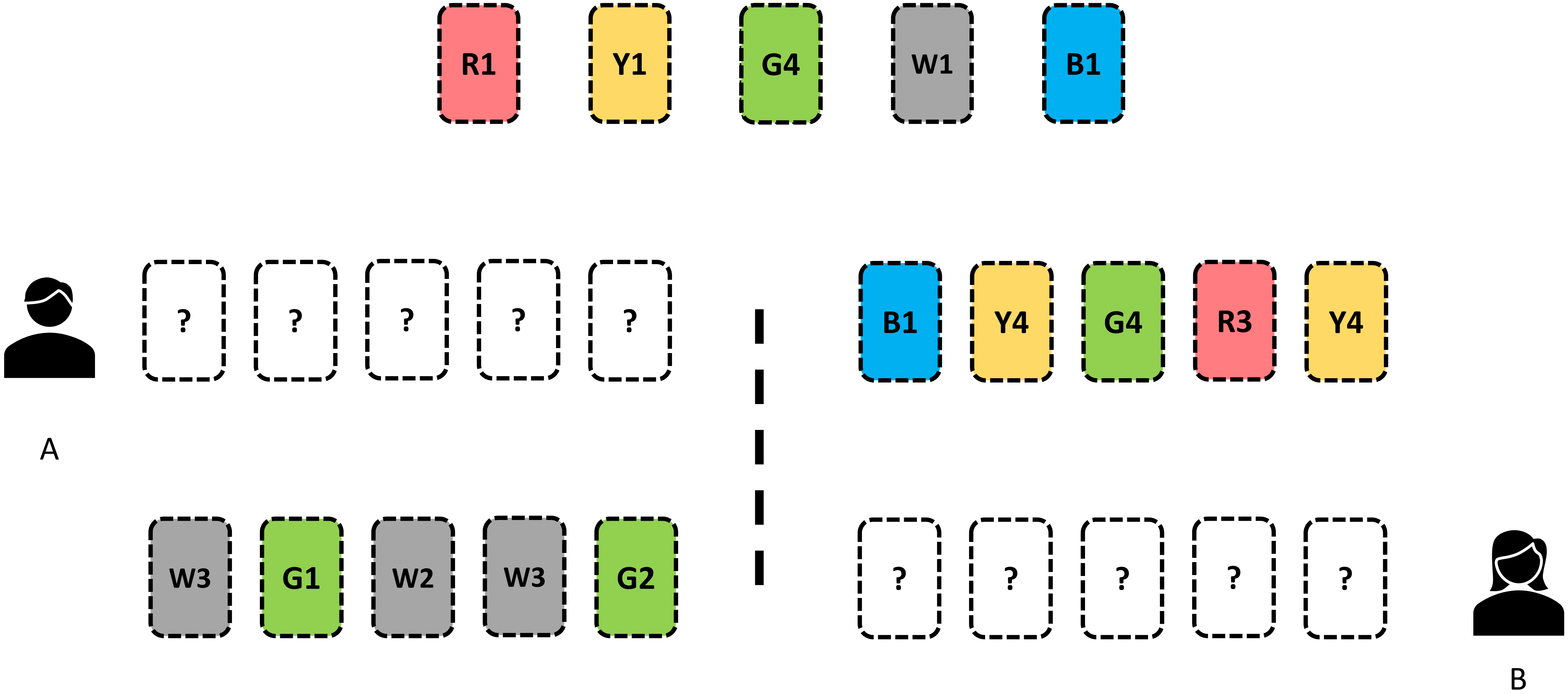
A

W3 G1 W2 W3 G2

? ? ? ? ?

B


R1
Y1
G3
W1
B1
?
?
G4
?
?
B1
Y4
G4
R3
Y4
A
W3
G1
W2
W3
G2
?
?
?
?
?
B



R1 Y1 G4 W1 B1

? ? ? ? ?

B1 Y4 G4 R3 Y4

A

W3 G1 W2 W3 G2

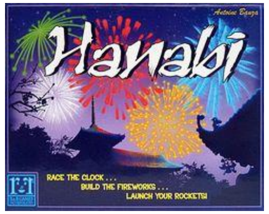
? ? ? ? ?

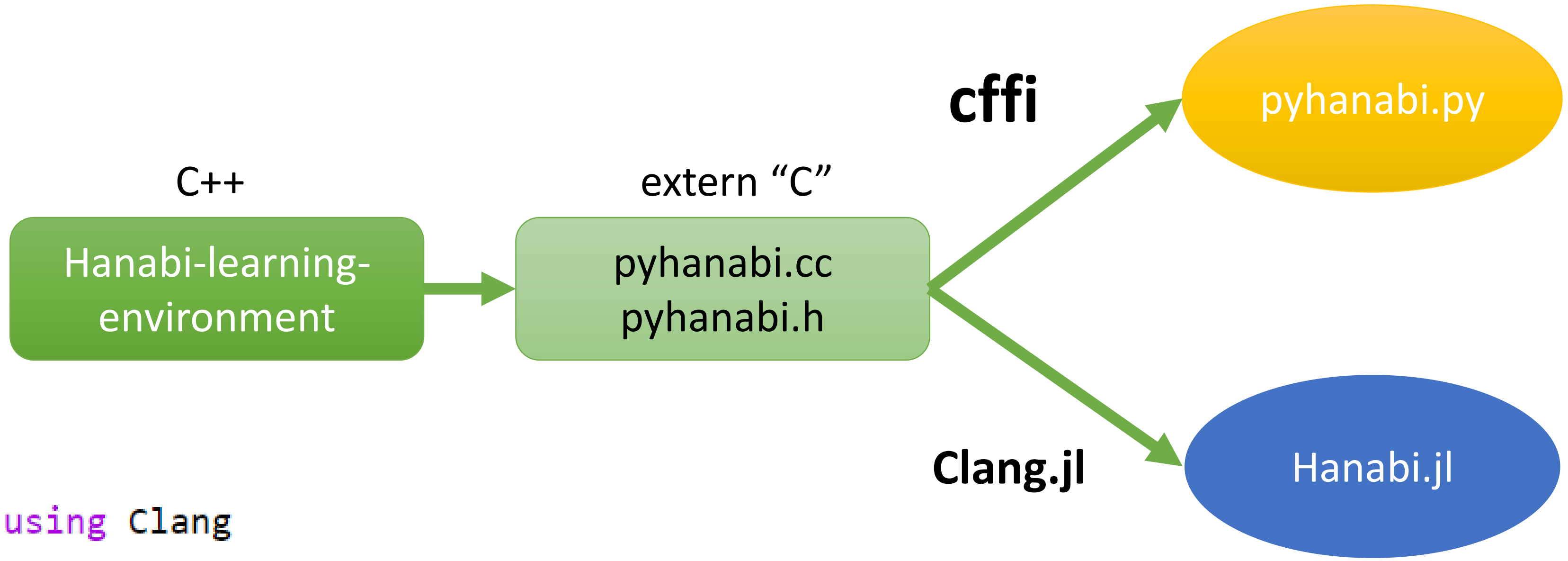
B

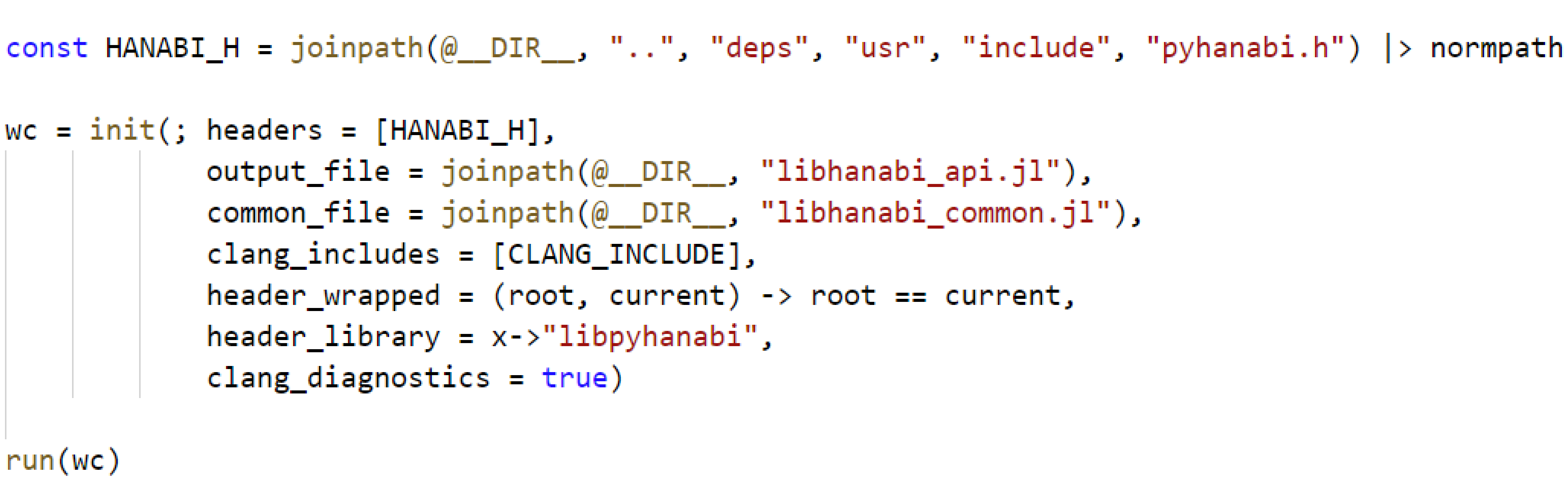
Write the engine in pure Julia

julia

Write a wrapper in pure Julia

	C/C++	ccall
ViZDOOM	C++	CxxWrap.jl
TextWorld	Python	PyCall.jl
……	……	……





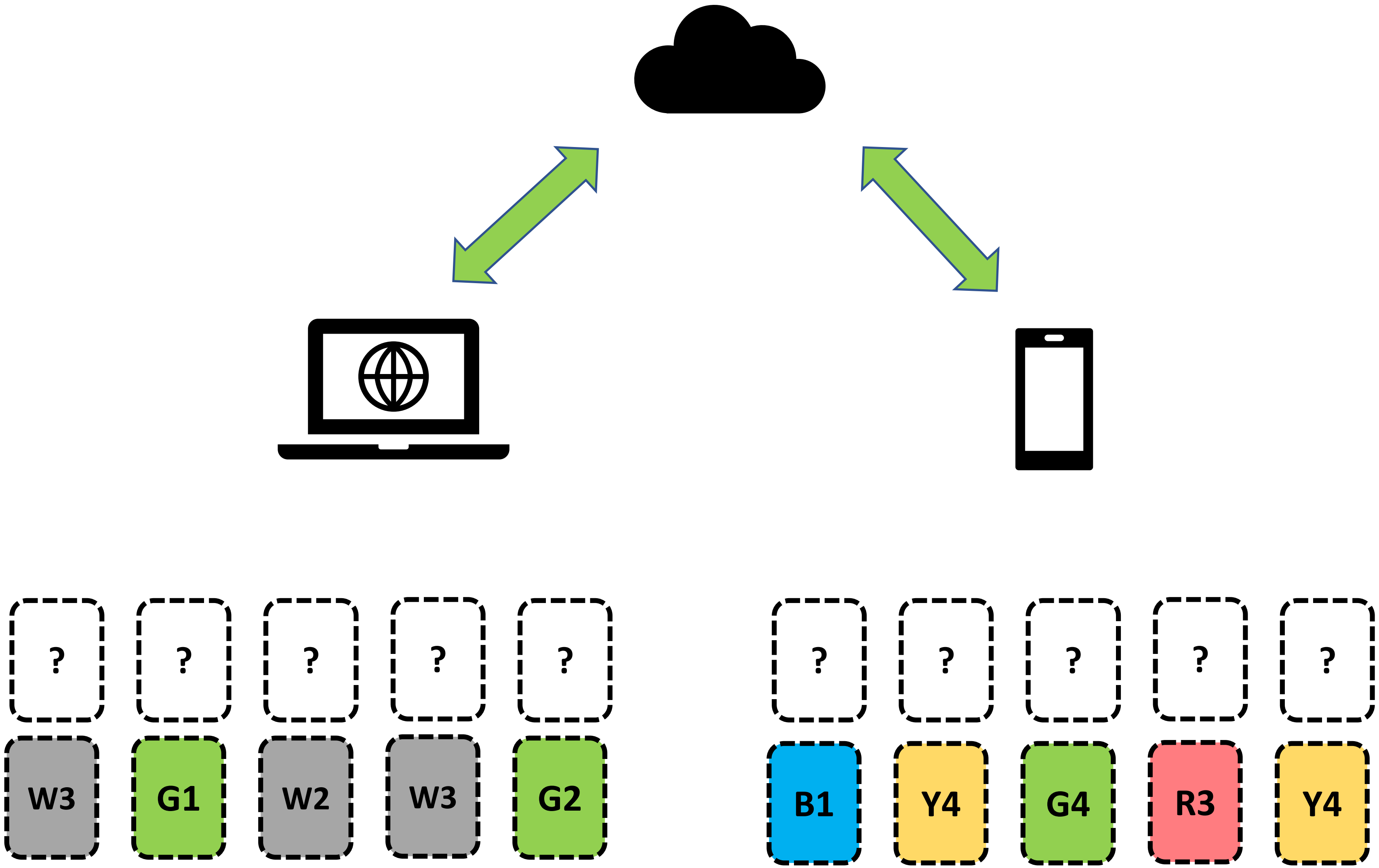
```julia
using Clang

const HANABI_H = joinpath(@__DIR__, "..", "deps", "usr", "include", "pyhanabi.h") |> normpath

wc = init(; headers = [HANABI_H],
            output_file = joinpath(@__DIR__, "libhanabi_api.jl"),
            common_file = joinpath(@__DIR__, "libhanabi_common.jl"),
            clang_includes = [CLANG_INCLUDE],
            header_wrapped = (root, current) -> root == current,
            header_library = x->"libpyhanabi",
            clang_diagnostics = true)

run(wc)
```

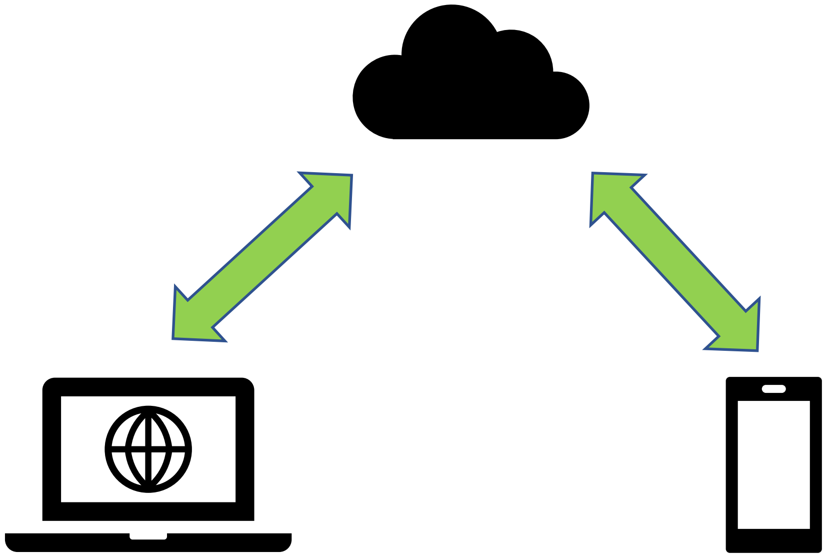
finalizer

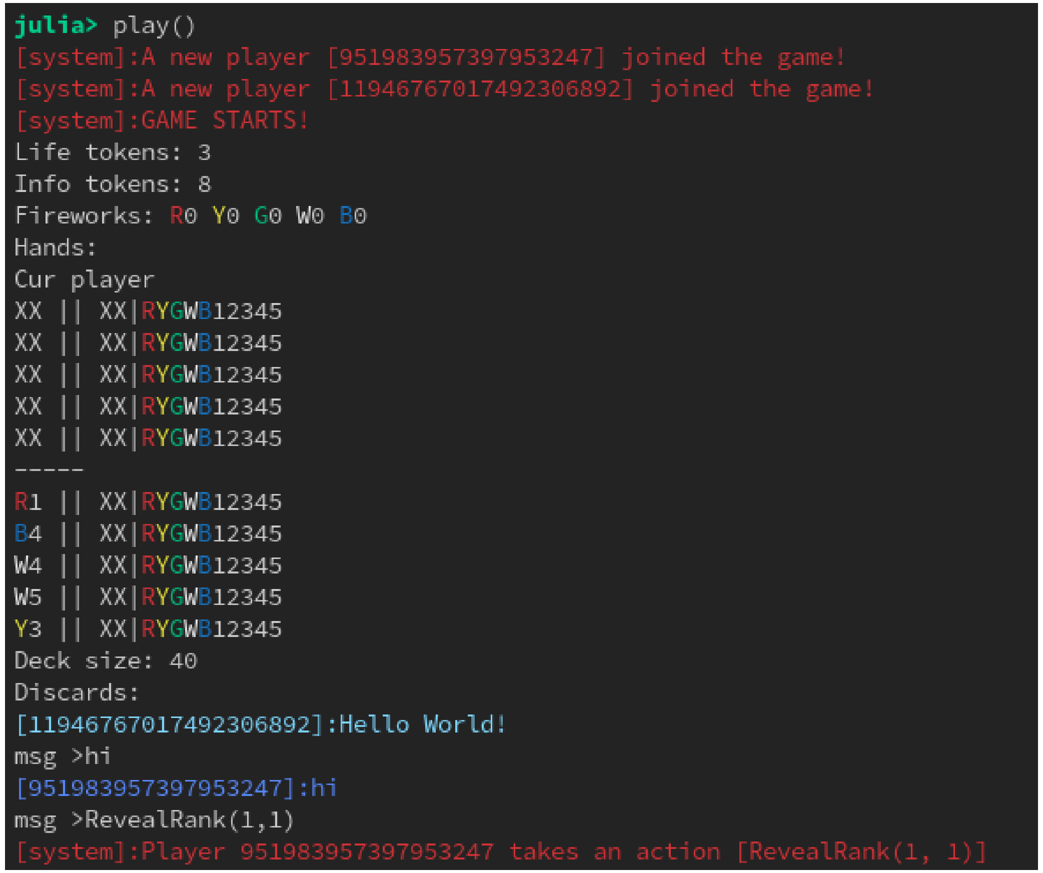


?	?	?	?	?
W3	G1	W2	W3	G2

?	?	?	?	?
B1	Y4	G4	R3	Y4

HTTP.WebSockets.WebSocket





```
julia> play()
[system]:A new player [951983957397953247] joined the game!
[system]:A new player [11946767017492306892] joined the game!
[system]:GAME STARTS!
Life tokens: 3
Info tokens: 8
Fireworks: R0 Y0 G0 W0 B0
Hands:
Cur player
XX || XX|RYGWB12345
XX || XX|RYGWB12345
XX || XX|RYGWB12345
XX || XX|RYGWB12345
XX || XX|RYGWB12345
-----
R1 || XX|RYGWB12345
B4 || XX|RYGWB12345
W4 || XX|RYGWB12345
W5 || XX|RYGWB12345
Y3 || XX|RYGWB12345
Deck size: 40
Discards:
[11946767017492306892]:Hello World!
msg >hi
[951983957397953247]:hi
msg >RevealRank(1,1)
[system]:Player 951983957397953247 takes an action [RevealRank(1, 1)]
```

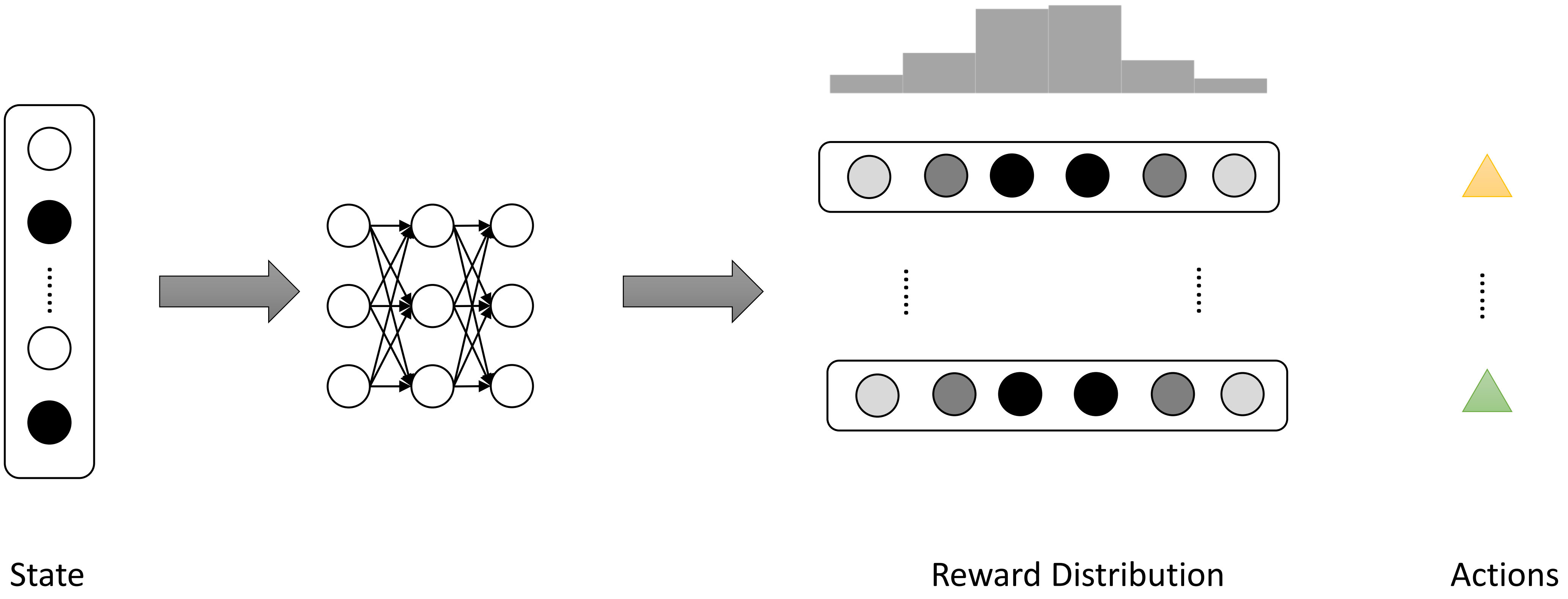
jupyter + Interact.jl

+ Microsoft Bot Framework

How to train a smart agent?

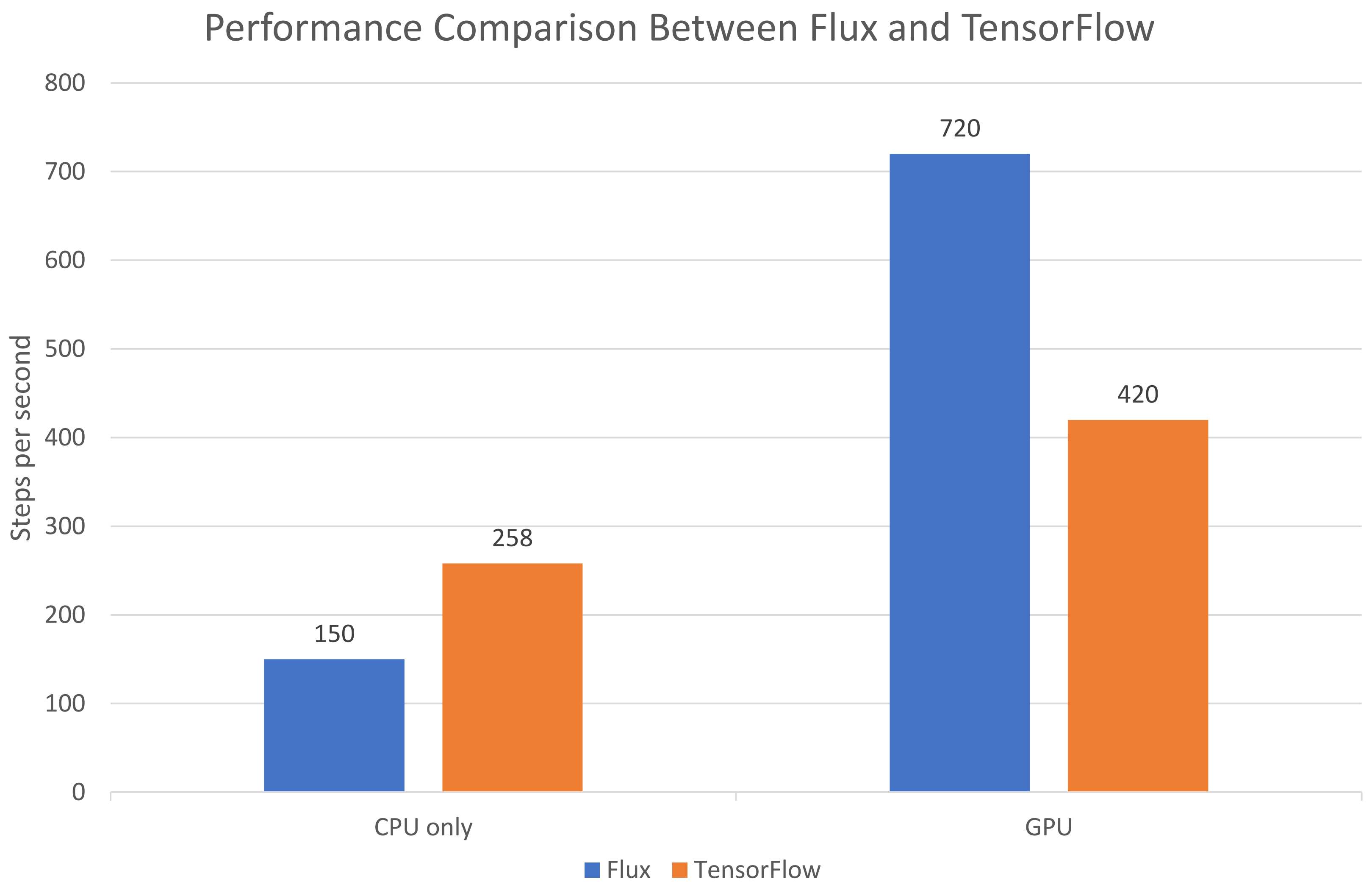
Rainbow

- **Distributional RL(C51)**
- **Prioritized Replay**
- **Multi-step Learning**
- Dueling Networks
- Double Q-learning
- Noise Nets



```
Chain(
        Dense(658, 512),
        Dense(512, 512),
        Dense(512, 20 * 51)
)
```

softmax



Environment:

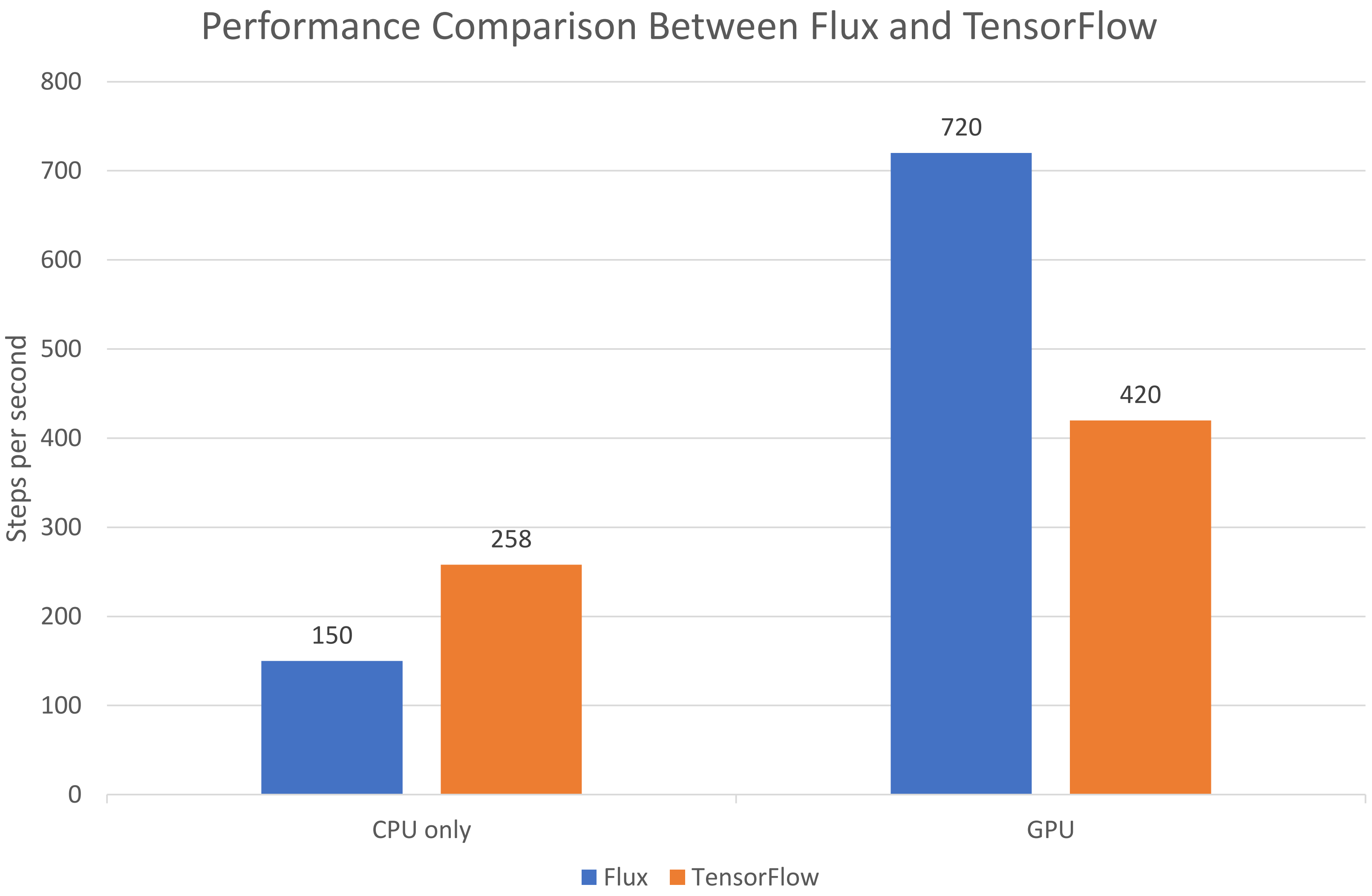
Azure Standard_NC12s_v3
V100 card
Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (×12)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

Julia v1.1.1
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1

Q1: Faster on GPU but slower on CPU?



Environment:

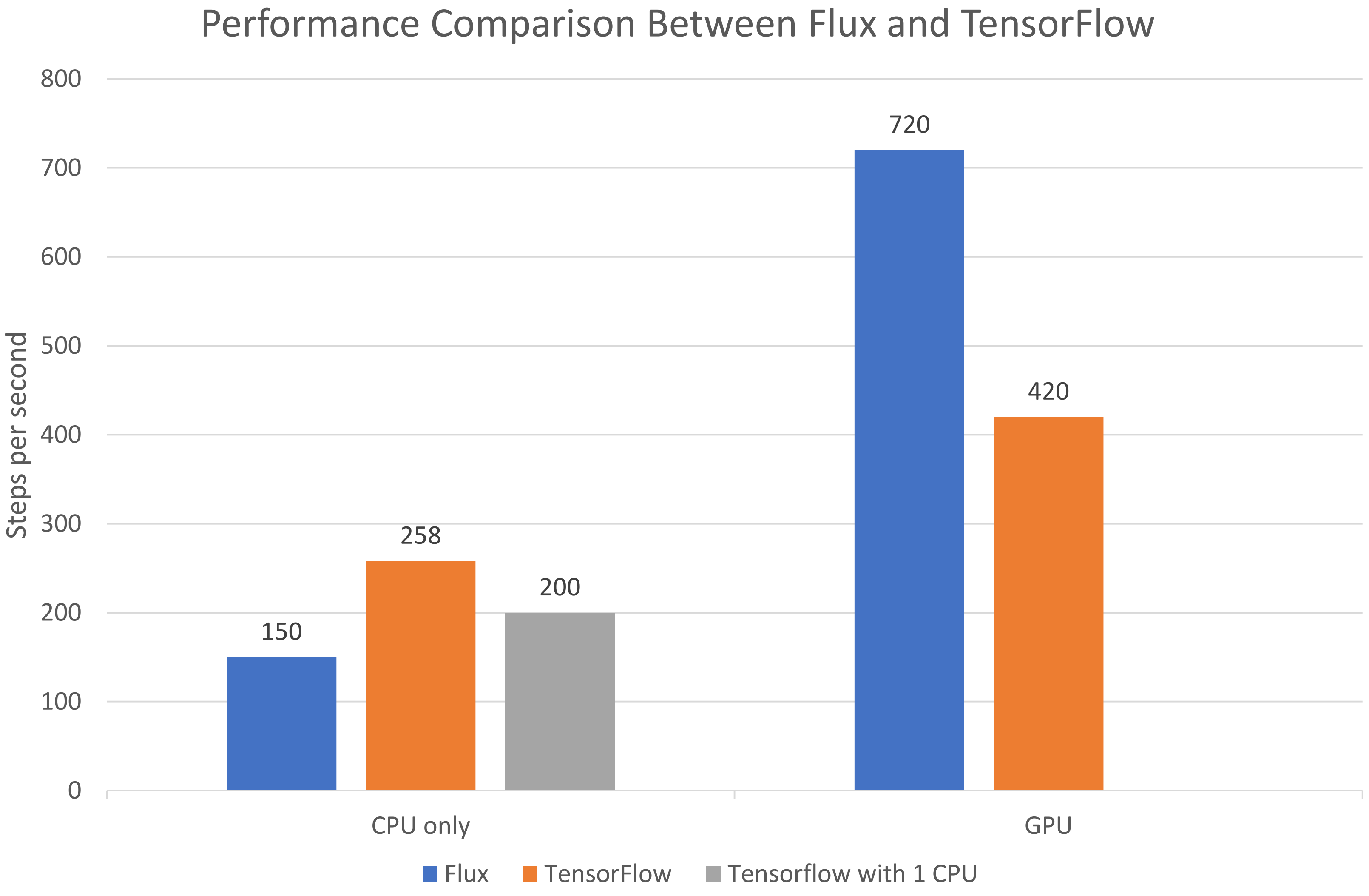
Azure Standard_NC12s_v3
V100 card
Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (×12)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

Julia v1.1.1
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1

Q1: Faster on GPU but Slower on CPU?



Environment:

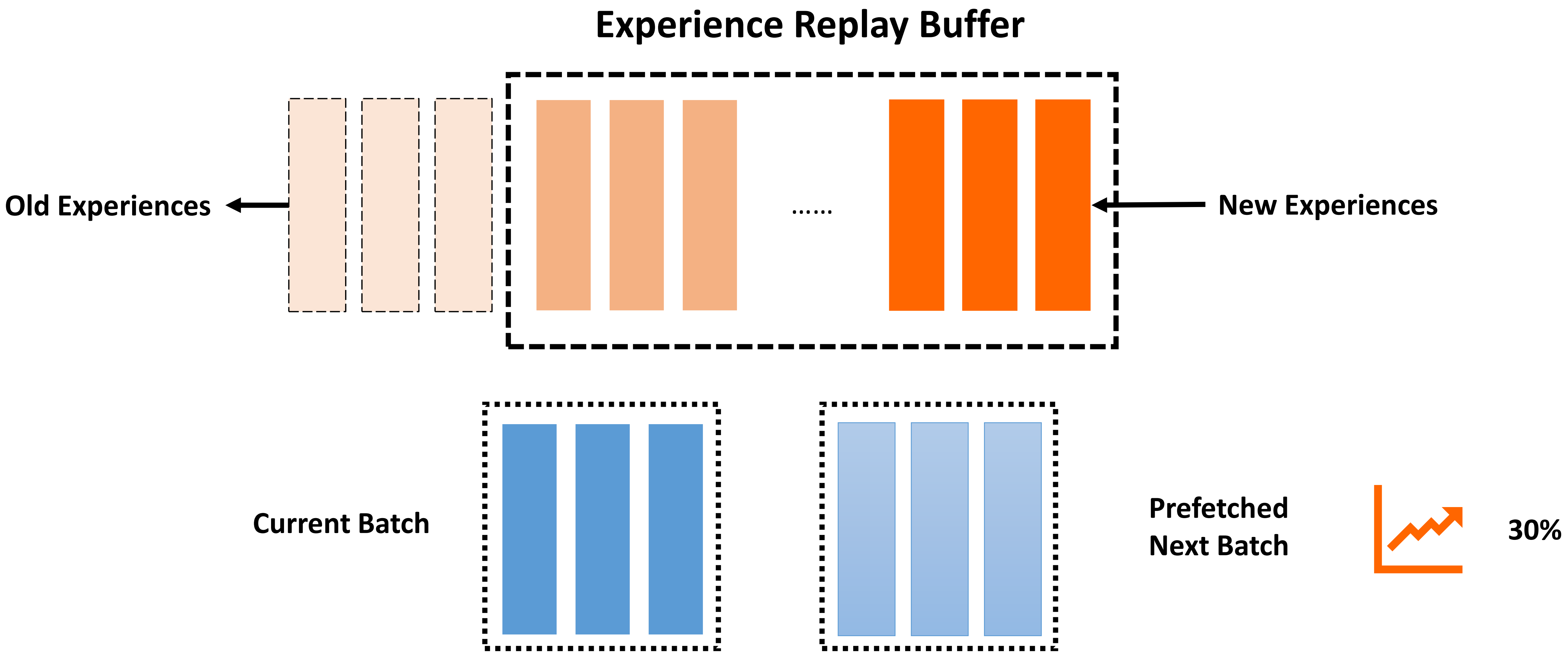
Azure Standard_NC12s_v3
V100 card
Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (×12)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

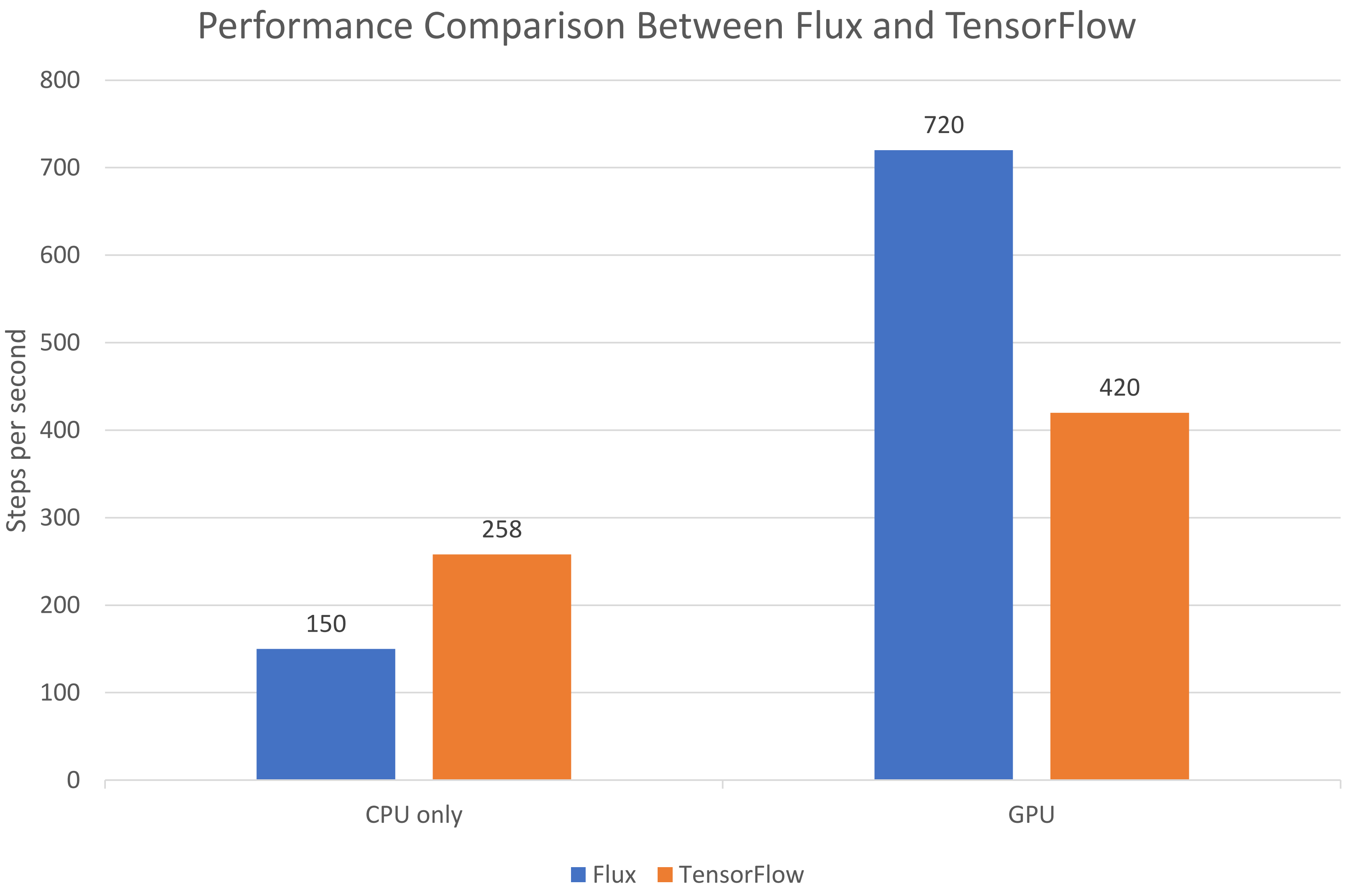
Julia v1.1.1
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1

tf.contrib.staging.StagingArea



Q2: Special Tricks on GPU?



Environment:

Azure Standard_NC12s_v3
V100 card
Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (×12)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

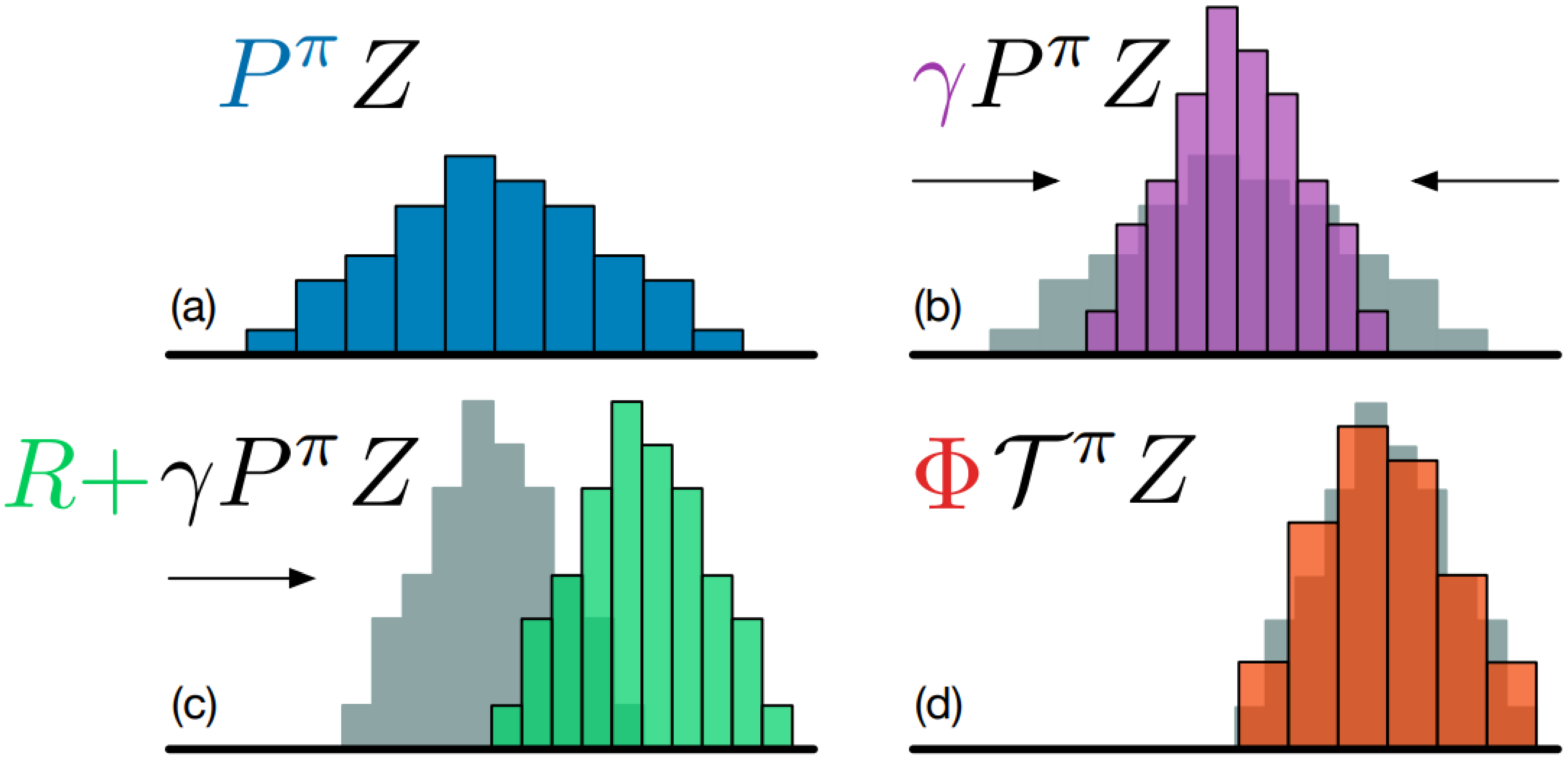
Julia v1.1.1
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1

Tricks:

```
# using CuArrays
```

It’s Just 3 Dense Layers Except …



Source: A Distributional Perspective on Reinforcement Learning (Figure 1)

Fused Broadcast

$$(\Phi\hat{\mathcal{T}}Z_\theta(x,a))_i = \sum_{j=0}^{N-1}\left[1-\frac{\left|[\hat{\mathcal{T}}z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}} - z_i\right|}{\triangle z}\right]_0^1 p_j(x',\pi(x'))$$

Python

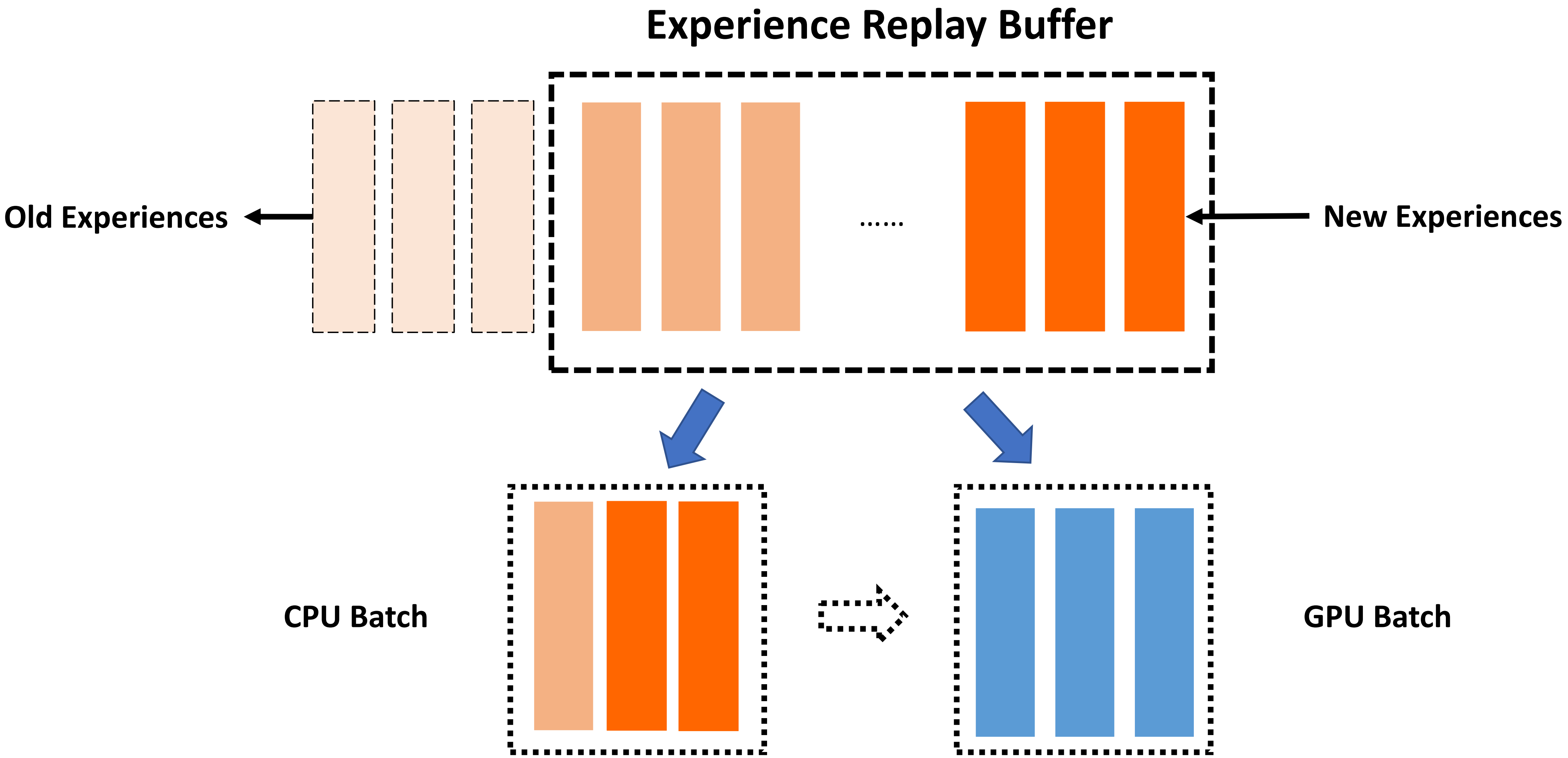
```
reshaped_target_support = tf.reshape(reshaped_target_support, [batch_size, num_dims, 1])
numerator = tf.abs(tiled_support - reshaped_target_support)
quotient = 1 - (numerator / delta_z)
clipped_quotient = tf.clip_by_value(quotient, 0, 1)
weights = weights[:, None, :]
inner_prod = clipped_quotient * weights
projection = tf.reduce_sum(inner_prod, 3)
```

I can turn on XLA!

Julia

```
clamp.(1 .- abs.(tiled_support .- target_support) ./ delta_z, 0, 1) .* weights
```

@views



Experience Replay Buffer

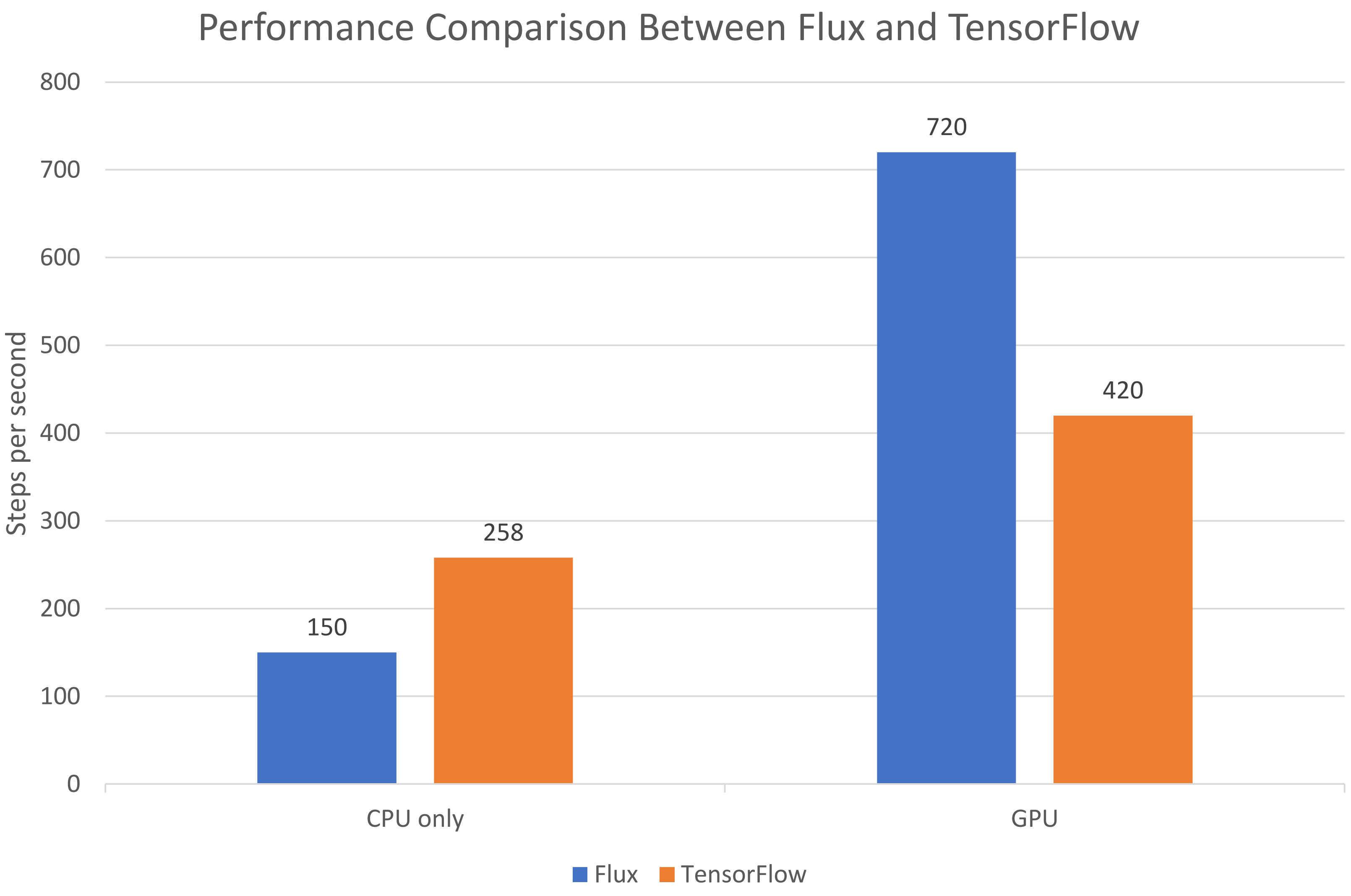
Old Experiences

New Experiences

CPU Batch

GPU Batch

Q3: The Impact of Hardware?



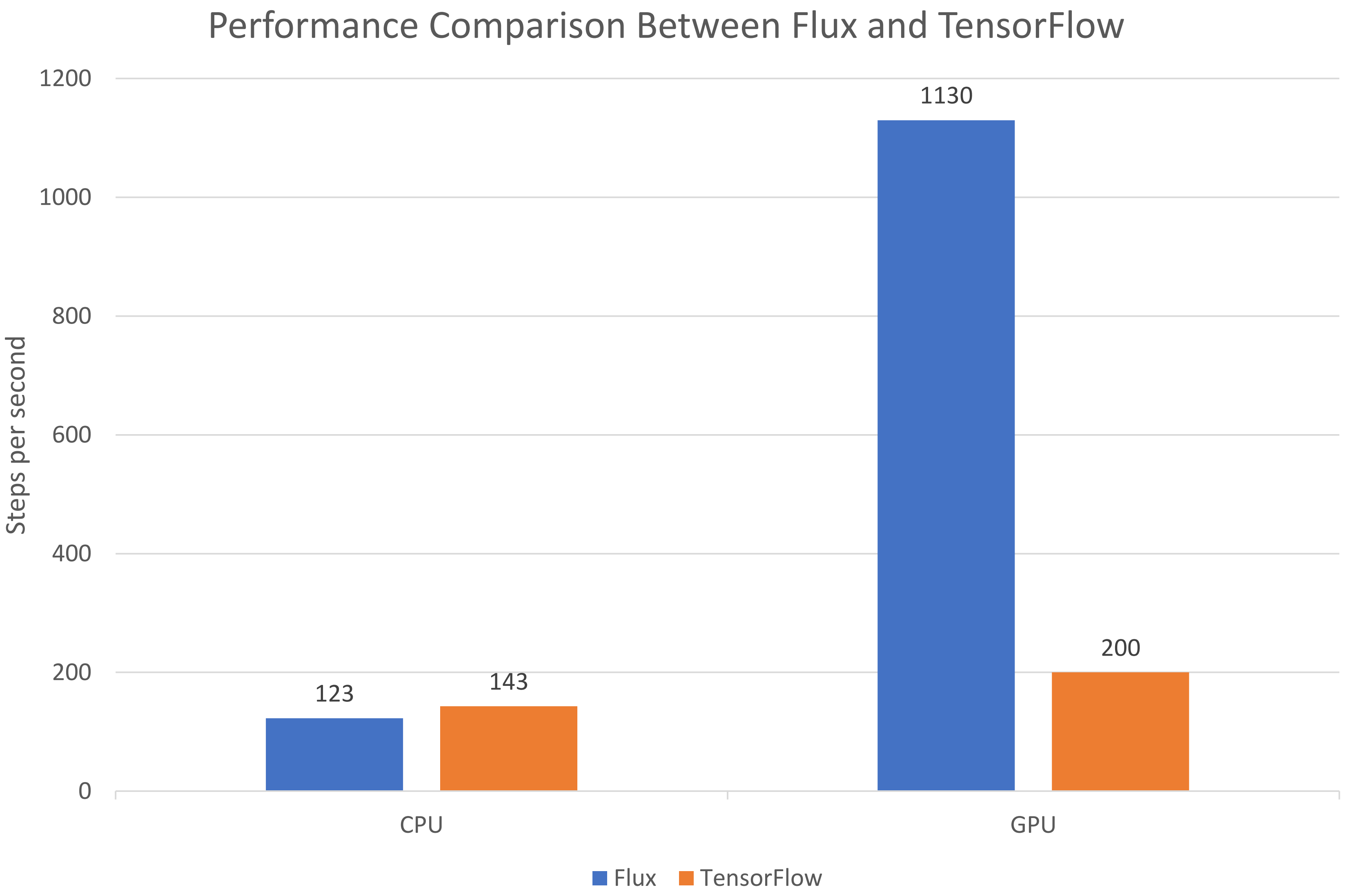
Environment:

Azure Standard_NC12s_v3
V100 card
Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz (×12)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

Julia v1.1.1
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1



Environment:

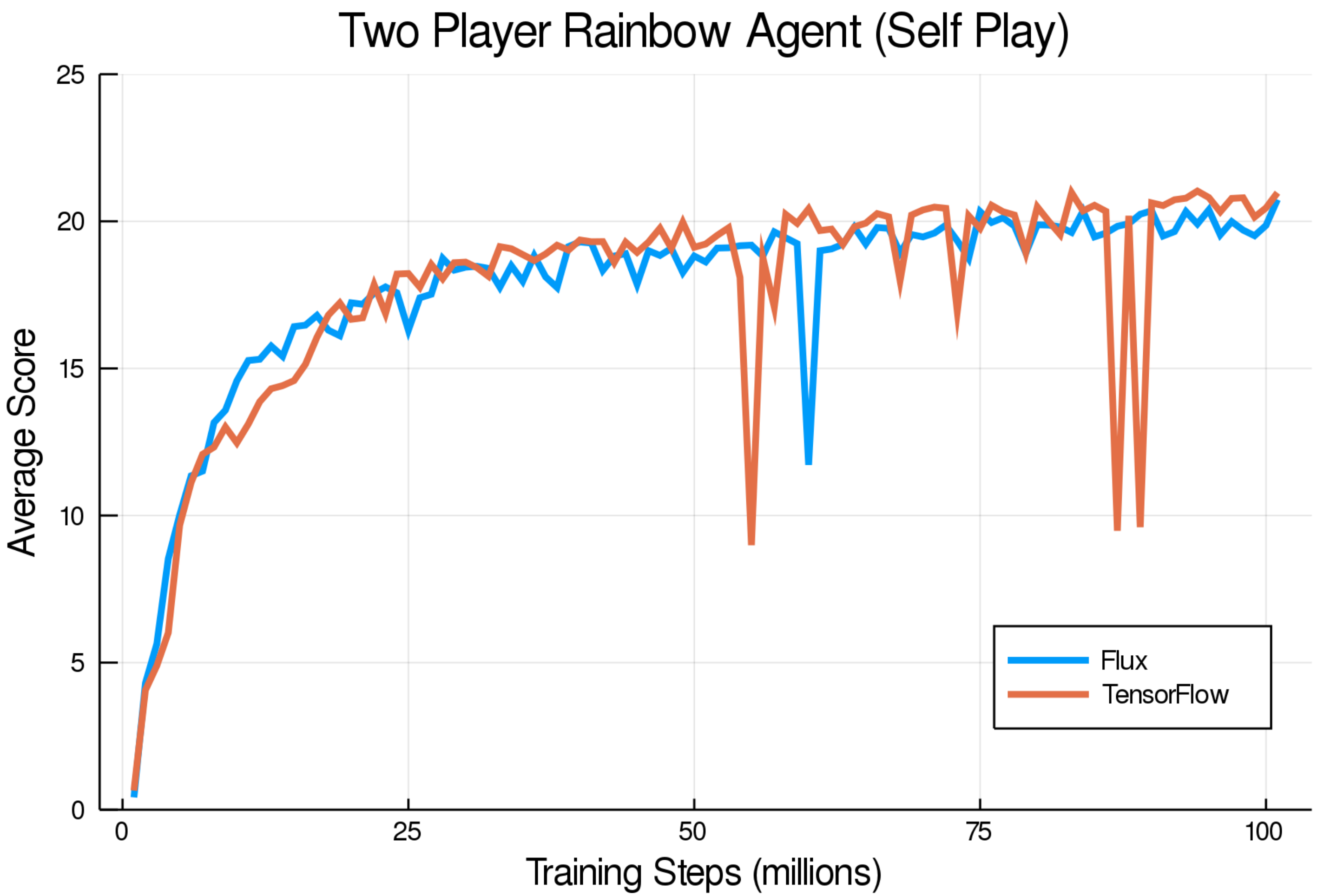
RTX 2080 TI card

Intel(R) Xeon(R) W-2123 CPU
@ 3.60GHz(×8)

Ubuntu 18.04
CUDA 10.0
CUDNN 7.6.0

Julia v1.1.0
Flux v0.8.3
Tracker v0.2.2

Python v3.7.3
TensorFlow v1.13.1



Training Time with 2080TI:

Flux: ≈ **1** day
TensorFlow: ≈ **5** days

"… I was wondering, … if you are willing to share it (pretrained model) with us. Our computation power is limited, and it would be very expensive for us training many agents…"

How about giving julia a try and get some free speedup?

Takeaway Messages

- Remember to release resources with a finalizer
- Avoid extra allocation with @view
- Fused broadcast will make your code fast and easy to read

Reference

For More Details:

- https://tianjun.me/essays/Lets_Play_Hanabi
    - Bayesian Action Decoder
    - Distributed Prioritized Experience Replay

Some Useful Links:

- https://github.com/JuliaReinforcementLearning/Hanabi.jl
- https://github.com/deepmind/hanabi-learning-environment
- https://arxiv.org/abs/1710.02298 (Rainbow)

- https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl
- https://github.com/Ju-jl/ReinforcementLearningAnIntroduction.jl

More Questions?

Special thanks to

- @Huda
- @Roger
- @Gnimuc
- @jbrea

for their guide and inspiration