Poster: Online RL in the Programmable Dataplane with OPaL

Kyle A. Simpson
University of Glasgow
Glasgow, Scotland
k.simpson.1@research.gla.ac.uk

Dimitrios P. Pezaros
University of Glasgow
Glasgow, Scotland
Dimitrios.Pezaros@gla.ac.uk

ABSTRACT

Reinforcement learning (RL) is a key tool in data-driven networking for learning to control systems online. While recent research has shown how to offload machine learning tasks to the dataplane (reducing processing latency), online learning remains an open challenge unless the model is moved back to a host CPU, harming latency-sensitive applications. Our poster introduces *OPaL*—On Path Learning—the first work to bring *online reinforcement learning* to the dataplane. OPaL makes online learning possible in SmartNIC/NPU hardware by returning to classical RL techniques—avoiding neural networks. This simplifies update logic, enabling online learning, and benefits well from the parallelism common to SmartNICs. We show that our implementation on Netronome SmartNIC hardware offers concrete latency improvements over host execution.

CCS CONCEPTS

• **Networks** → **Data path algorithms**; **Programmable networks**; • **Computing methodologies** → **Reinforcement learning**.

1 INTRODUCTION

Automatic network optimisation, control, and defence are becoming commonplace through data-driven techniques such as RL methods, where every change and its effects improve future decisions made by an agent. In tandem, P4 [1] and *programmable dataplane* hardware have inspired explosive growth and interest in the research community surrounding in-network computation and offloading. This ecosystem exposes not only runtime reconfigurable packet processing, but per-packet and per-flow traffic measurement state that can greatly aid network operation [2].

Machine learning in the dataplane: To process this fine-grained state both at line rate *and* at low latency, there has been keen interest in executing ML models in the dataplane [4, 6]. Works in this category, while powerful, operate by converting a *pre-trained* model into a suitable representation, such as a binary neural network or string of match-action tables. These can react to on-device state, yet the missing piece of the puzzle is learning and updating these ML analyses *online*, without deferring to another machine in the network. This lacuna has yet to be addressed by the community.

Problems of online training: Programmable dataplane hardware, being designed solely for efficient packet processing, lacks floating-point arithmetic support, even in the case of more general purpose NPU-type SmartNICs. Additionally, deep neural network training relies on backpropagation, can require memorizing many sizeable replay buffers for RL, and needs many mini-batches of data for stable training. The natural solution is to pass training data to a host machine or dedicated accelerator, yet this adds delays in crossing the PCIe bus, kernel-user handover, and buffering—adding latency while learning. The above (NIC-suitable) inference schemes still must be trained offline, even though their data formats solve the issue of floating point use. Online learning thus requires careful choices in sample data formats and the learned function approximation scheme.

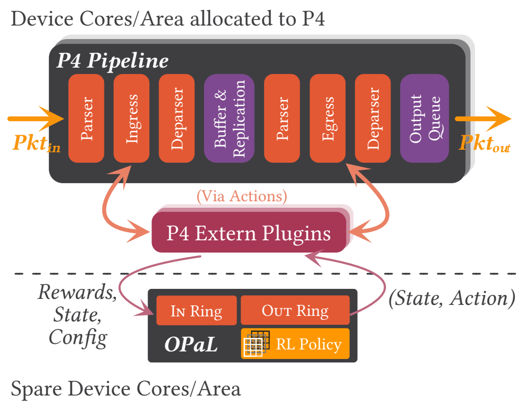


Figure 1: OPaL brings low-latency, online reinforcement learning to SoC- and NPU-based SmartNICs. Classical RL policy methods are the key to making this computationally feasible.

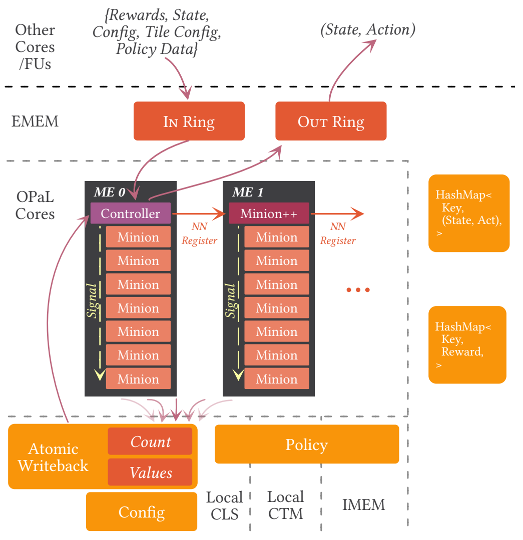


Figure 2: Parallel policy execution. A single *controller* delegates RL computation and updates to many *minion* threads, using a shared atomic writeback.

2 OPAL'S DESIGN

OPaL makes RL workloads feasible on SmartNICs by using quantised fixed-point representations of action values, tile-coded policies, and one-step temporal-difference RL algorithms such as Sarsa [5].

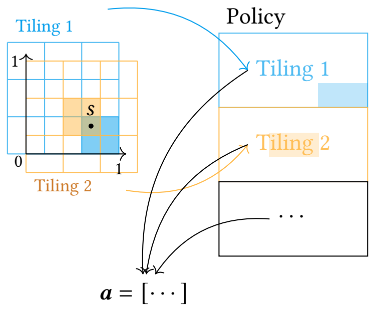


Figure 3: Tile-coding: actions preferences are aggregated from *disjoint* tile queries—a map-reduce problem. To update, gradients are simply the tiles activated during the forward pass with no aggregation.

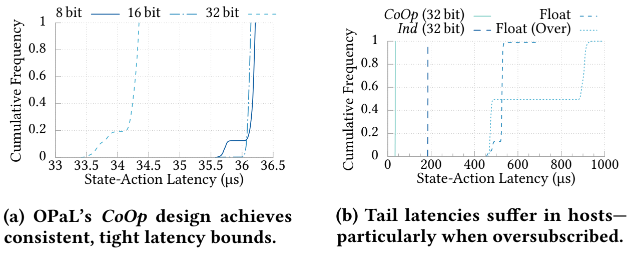
These allow us to evaluate and update policies using *only integer arithmetic*, and are computationally simple. We observe that tile coded inference is a map-reduce problem (fig. 3), and thus over integers admits a novel *wait-free Sarsa* RL algorithm (exploiting the parallelism built in to SmartNICs). While these functions have lower theoretical model capacity, they do not require batches of inputs to learn in a stable way, negating the memory needed to store experience replays or minibatches.

Interacting with OPaL: To prevent packet stalls but maintain access to fine-grained state, OPaL places RL execution on-NIC, *but off the main packet path*, parallel to the main dataplane (fig. 1). The P4 (or device-specific) dataplane pushes state vectors and reward measurements to OPaL's In ring, and pulls output actions when it is able to, imposing minimal impact on carried traffic for both bump-in-the-wire deployments and at end-points. Once an action is enqueued on its Out ring, OPaL updates its policy.

Inference/Updates: State is sent to a set of packet processing threads, which compute hit tiles from a locally held work set. During inference, these threads atomically add values to a shared action preference list, while tile-coding guarantees that policy updates proceed without data races (fig. 2). In the case of our Netronome implementation, these run on lightweight cores known as *microengines* (MEs)—which could be substituted for suitable FUs on CGRA/FPGA hardware. Additionally, we have a single-threaded implementation, *Ind*, which forgoes this cooperative model to maximise offline throughput (by having many parallel single-threaded agents).

3 EVALUATION

We compare our implementation of OPaL against a tile-coded RL agent written in numpy on a powerful host (i7-6700K@4×4.2 GHz), using the policy dimensions of existing DDoS prevention work [3], across various quantisation bit depths. Figure 4 shows that our main design (*CoOp*) achieves order-of-magnitude latency improvements over our single-threaded variant (*Ind*) and a host implementation, as well as considerably tighter tail behaviour on the SmartNIC. Note that lower bit depths have higher execution costs due to the native 32 bit width of Netronome registers. On update times, we see 63.12 µs at 99.99th percentile for a 32 bit policy of this size. We relate a 15 × latency (*CoOp*) and 2.82 × offline throughput-per-core (*Ind*) improvement in spite of the Netronome's slower clock speed (0.29 ×), showing the value of exploiting SmartNIC parallelism.



(a) OPaL's *CoOp* design achieves consistent, tight latency bounds.

(b) Tail latencies suffer in hosts—particularly when oversubscribed.

Figure 4: Cumulative state-action latency plots for OPaL and host-based execution at different quantisation settings.

4 NEXT STEPS

We intend to measure the effects of our data format and function approximation choices on overall accuracy and convergence times for synthetic data, comparing different bit depths against tile-coded floating-point numbers and more expressive functions such as neural networks. Moreover, we intend to build OPaL into ultra low latency control scenarios, such as short-flow prioritisation or active queue management.

REFERENCES

[1] Pat Bosshart *et al.* 'P4: programming protocol-independent packet processors'. In: *Computer Communication Review* 44.3 (2014), pp. 87–95.
[2] Mojgan Ghasemi *et al.* 'Dapper: Data Plane Performance Diagnosis of TCP'. In: *Proceedings of the Symposium on SDN Research, SOSR 2017, Santa Clara, CA, USA, April 3-4, 2017.* ACM, 2017, pp. 61–74.
[3] Kyle A. Simpson *et al.* 'Per-Host DDoS Mitigation by Direct-Control Reinforcement Learning'. In: *IEEE Trans. Network and Service Management* 17.1 (2020), pp. 103–117.
[4] Giuseppe Siracusano *et al.* 'Running Neural Networks on the NIC'. In: *CoRR* abs/2009.02353 (2020). arXiv: 2009.02353.
[5] Richard S. Sutton *et al.* *Reinforcement Learning: An Introduction.* 2nd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, Nov. 2018. ISBN: 9780262039246.
[6] Zhaoqi Xiong *et al.* 'Do Switches Dream of Machine Learning?: Toward In-Network Classification'. In: *Proceedings of the 18th ACM Workshop on Hot Topics in Networks, HotNets 2019, Princeton, NJ, USA, November 13-15, 2019.* ACM, 2019, pp. 25–33.