KMLib: Towards Machine Learning For Operating Systems and Storage Components

Ibrahim Umit Akgun, Ali Selman Aydin, and Erez Zadok
Department of Computer Science, Stony Brook University

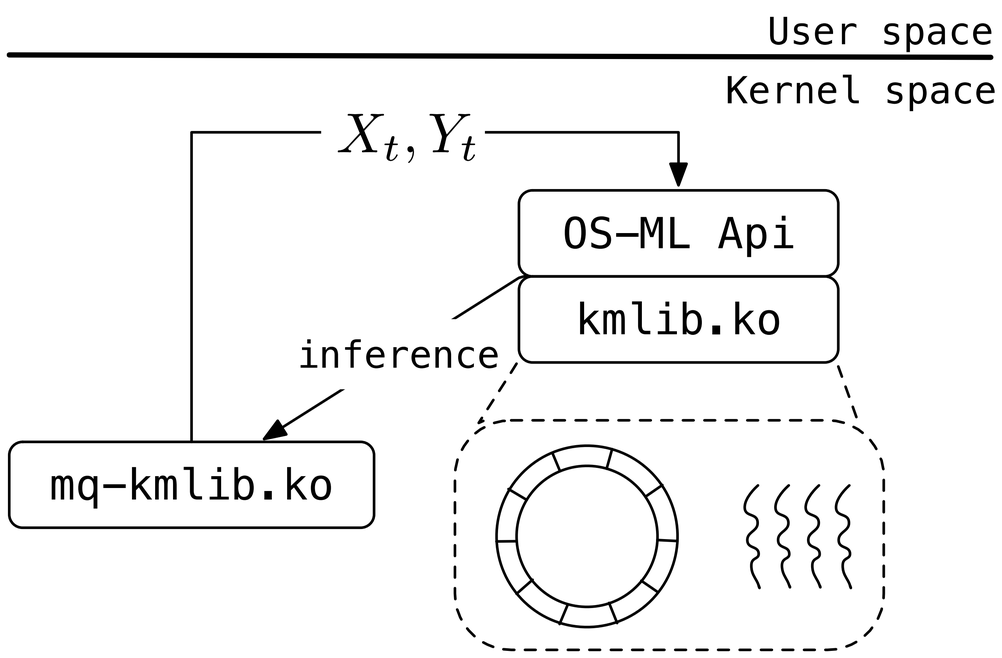
Motivation & Challenges ①

- **Motivations**
  - Adaptive systems ← data patterns and OS events
  - User-level ML engines are often too costly
  - A lightweight yet efficient ML engine → OS kernel
- **Challenges**
  - Extensive kernel programming skills
  - Debugging and fine-tuning ML models
  - Avoiding frequent user-kernel switches.

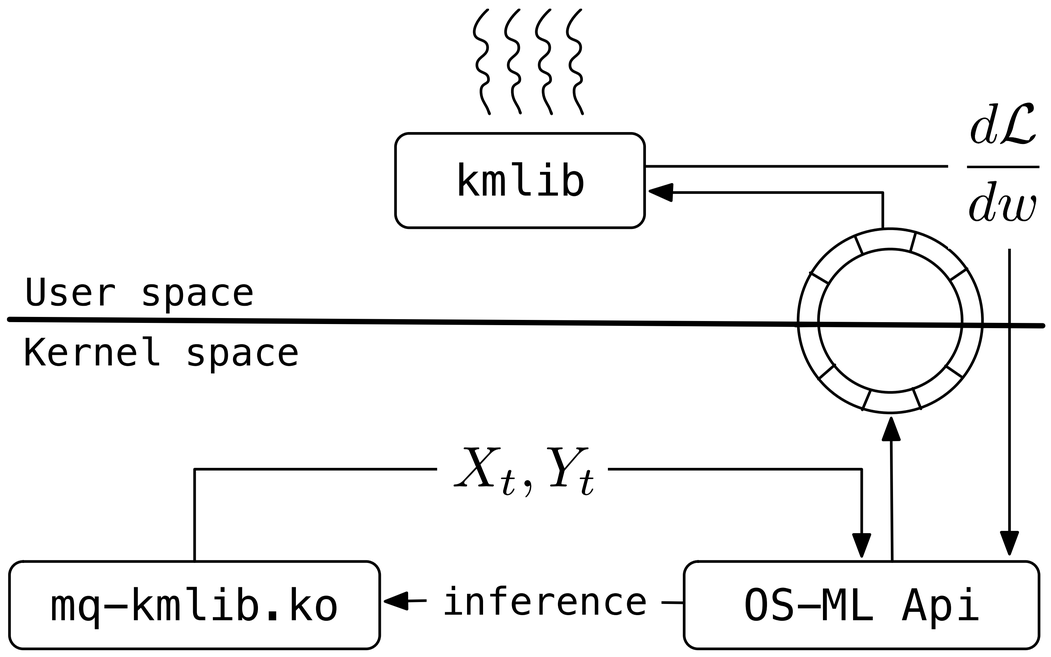
Machine learning library design ②

1. Support standard math floating-point functions in the kernel
2. Tensor-like representation for matrices and model parameters.
   - Adaptable forward and backprop; lock free d-s; parallelism
3. Adapt to new Workloads
   - few-shot learning[1], active learning[2]

Operating System Integration ③



Kernel Library



User-Kernel shared Library

User space vs. kernel space ④

- Offloading training and inference (sub μs level)
- User-kernel memory mapped shared mode
  - Collects data from the kernel space
  - Trains using user-space threads
  - Inference runs in kernel space ↓ $latency$
- User-kernel shared lock-free circular buffers[3]
- Easier developing, debugging, testing

Reducing computation & memory overheads

Computation and memory capping ⑤

- Offloads the training to library threads saving the input data and the predictions for training
  - Blocking mode process every single input data
    - Freq. of computation requests is high ↑ $overhead$
  - Dropping mode overruns unprocessed input data
    - May hurt training quality ↓ $overhead$

Low Precision Training

- x86 floating-point **kernel_fpu_begin.**
  - context-switch ↑ $overhead$

Evaluation ⑥

- Fine-tune *mq-deadline* I/O scheduler
- To predict whether the I/O request will meet deadline
- The regression model predicts issue time for a given I/O
  - Normalized block number & Ordinalized operation
- Predict with an accuracy of **74.62%**
  - Reduced the overall I/O latency by **8%**.
- Tests on QEMU with synthetic workloads
- We wrote nearly 3,000 lines of C/C++ code (LoC).
- User-space library → 96KB Kernel module → 804KB

References

[1] Wang, Y. and Yao, Q. Few-shot learning: A survey. arXiv preprint arXiv:1904.05046, 2019.
[2] Settles, B. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.
[3] Desnoyers, M. and Dagenais, M. R. Lockless multi-core high-throughput buffering scheme for kernel tracing. Operating Systems Review, 46(3):65–81, 2012.