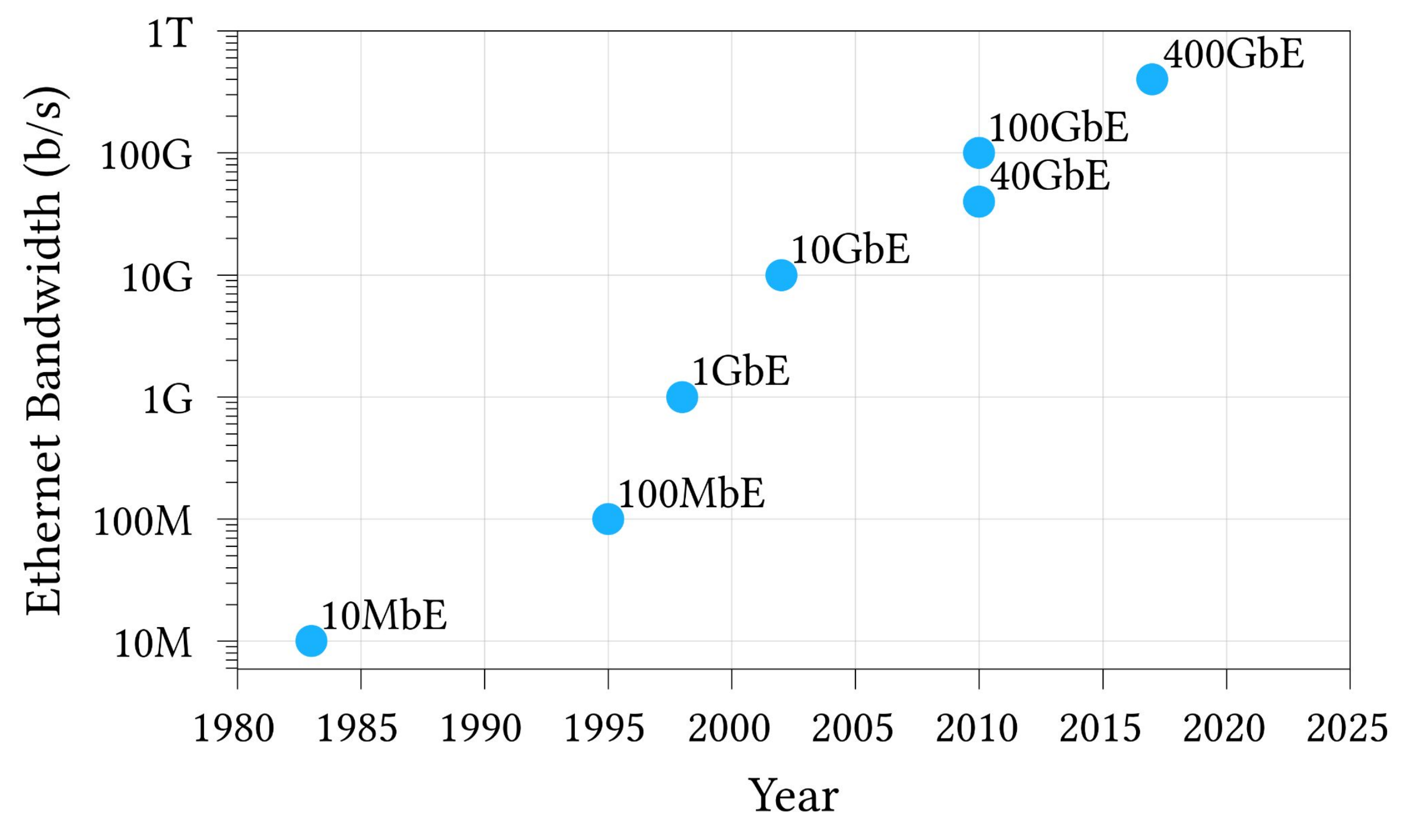
FlexMock: Restoring Development Flexibility when using SmartNICs

Fabian Mkocheko,[1] Hugo Sadok,[2] Justine Sherry[2]
[1]Middlebury College [2]Carnegie Mellon University

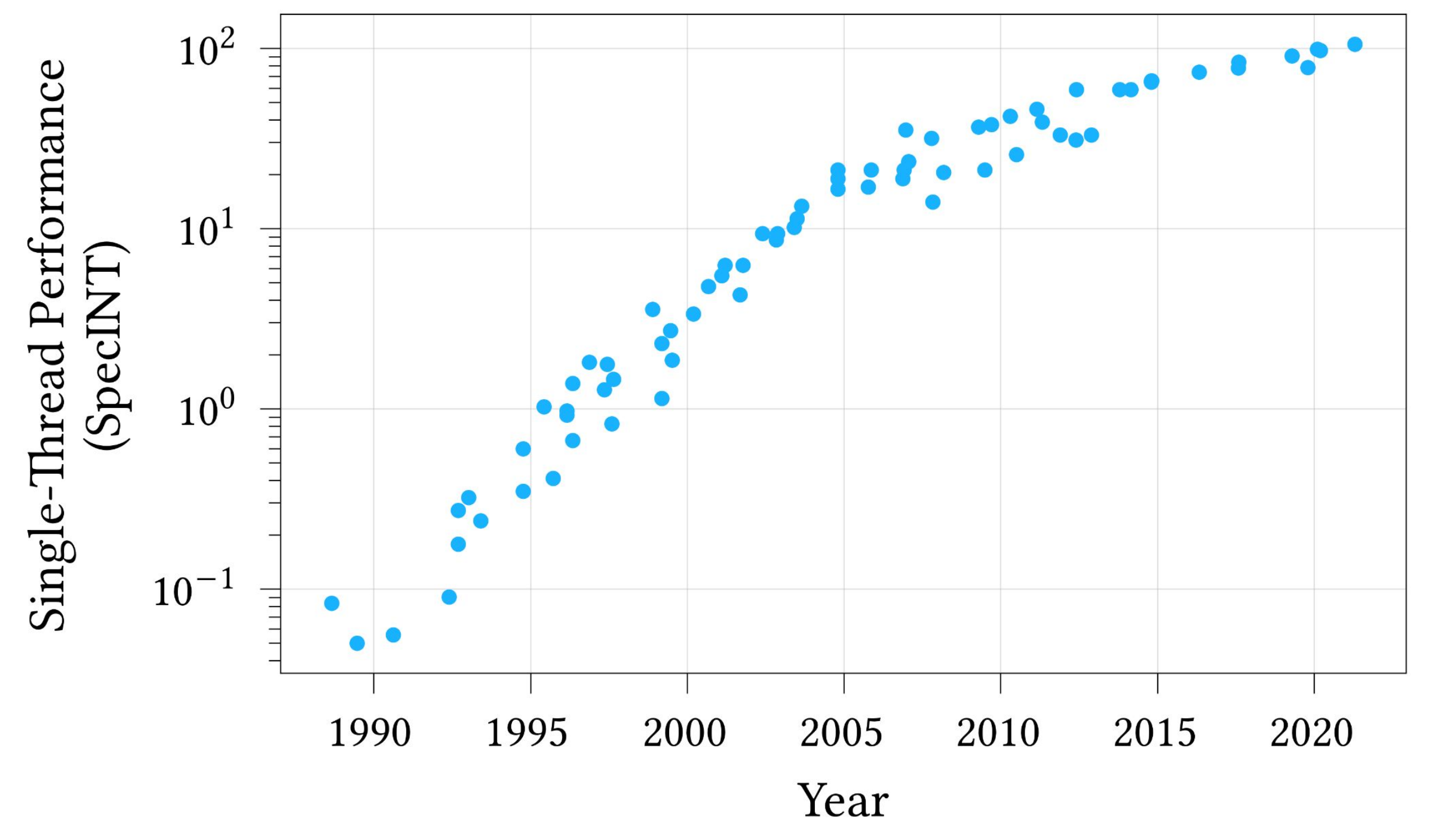
Background and Motivation

While link speeds continue to increase exponentially, CPU performance is plateauing.

Evolution of Network Link Speeds



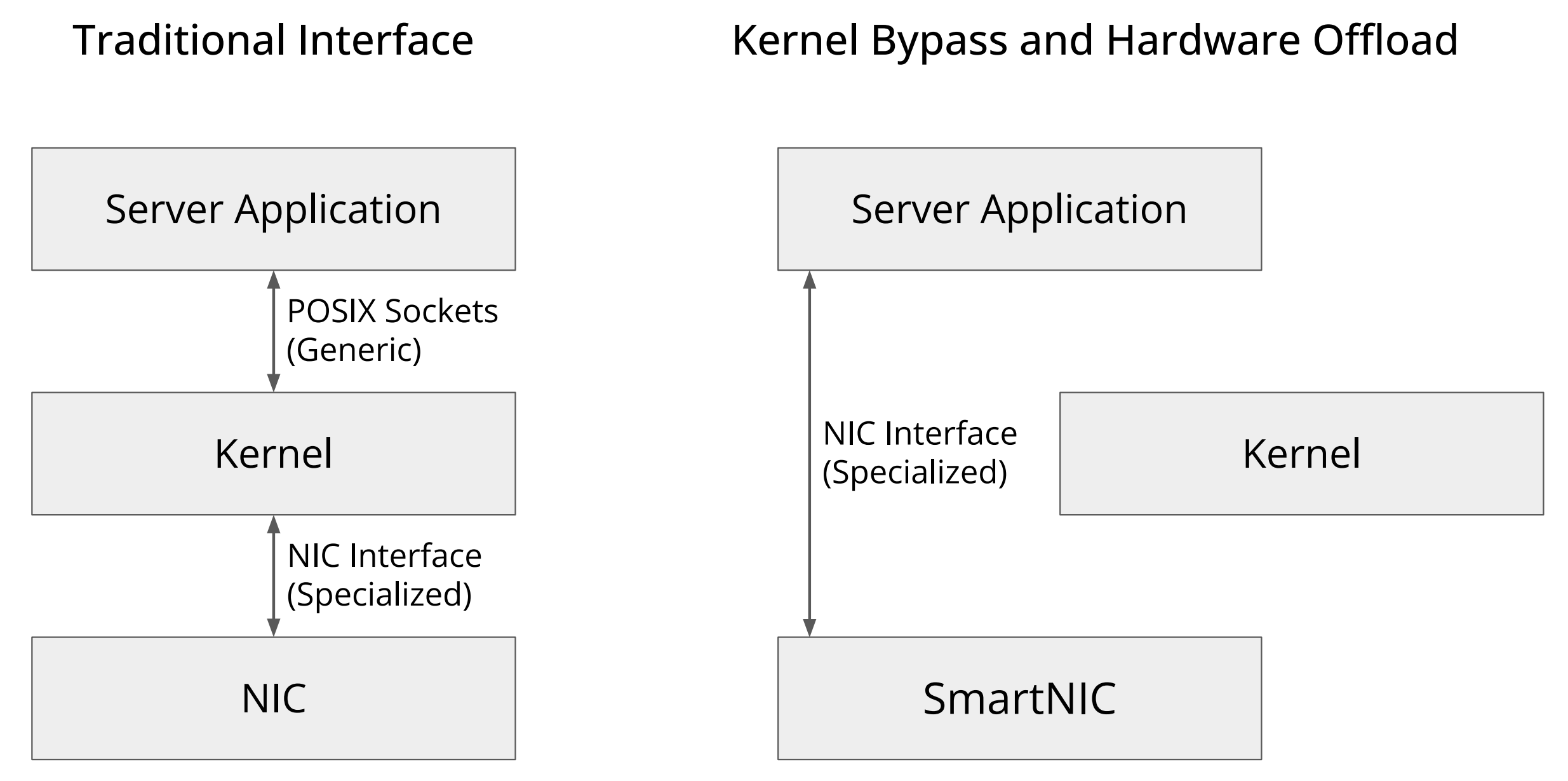
Evolution of CPU Single-Thread Performance



Data obtained from K. Rupp's "Microprocessor Trend Data"

Alternative Network Stack Interfaces

Network-intensive applications (e.g., key-value stores, network load balancers, authentication servers) now often bypass the kernel network stack and access the NIC directly.



SmartNICs allow hardware acceleration of the network stack.

Problem

The need for specialized hardware complicates application development.

Setup. Specialized hardware often involves complicated hardware and software setup, reducing developers' productivity.

Hardware access. Functionality testing and performance tuning can only be done by those with access to niche and expensive specialized hardware.

FlexMock Design

Key Idea: *The specificities of the specialized hardware can be abstracted away during development.*

FlexMock replicates the SmartNIC interface in software so that applications can still run when the hardware is unavailable.

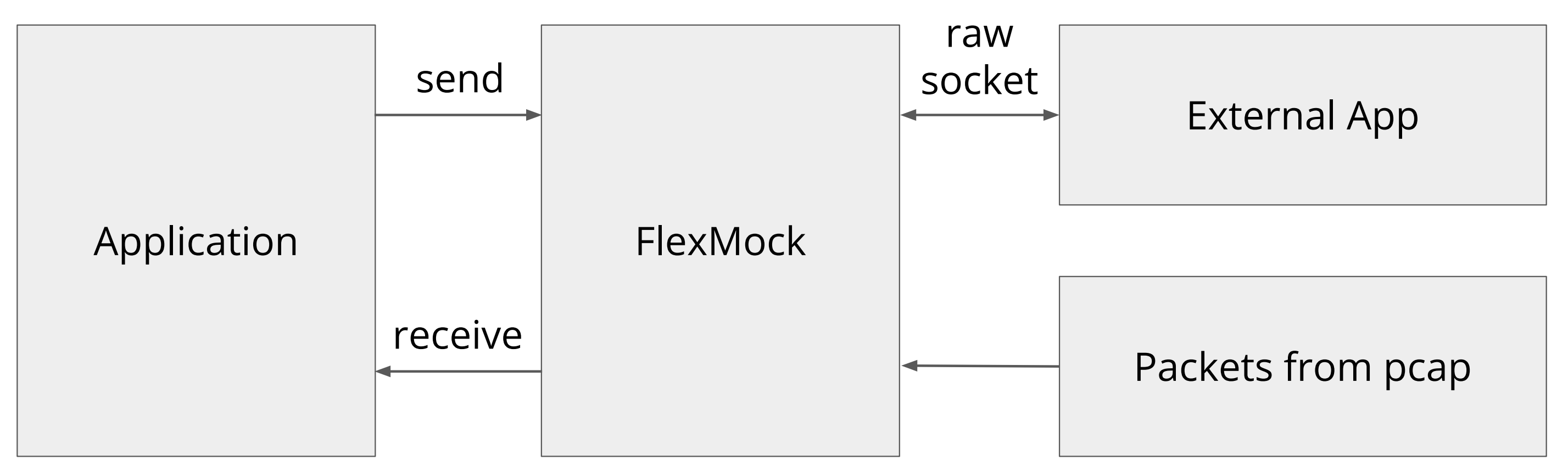
Goals

Performance tuning. FlexMock should allow developers to tune application performance without using the specialized hardware.

Functionality evaluation. FlexMock should let developers verify their application logic without using specialized hardware.

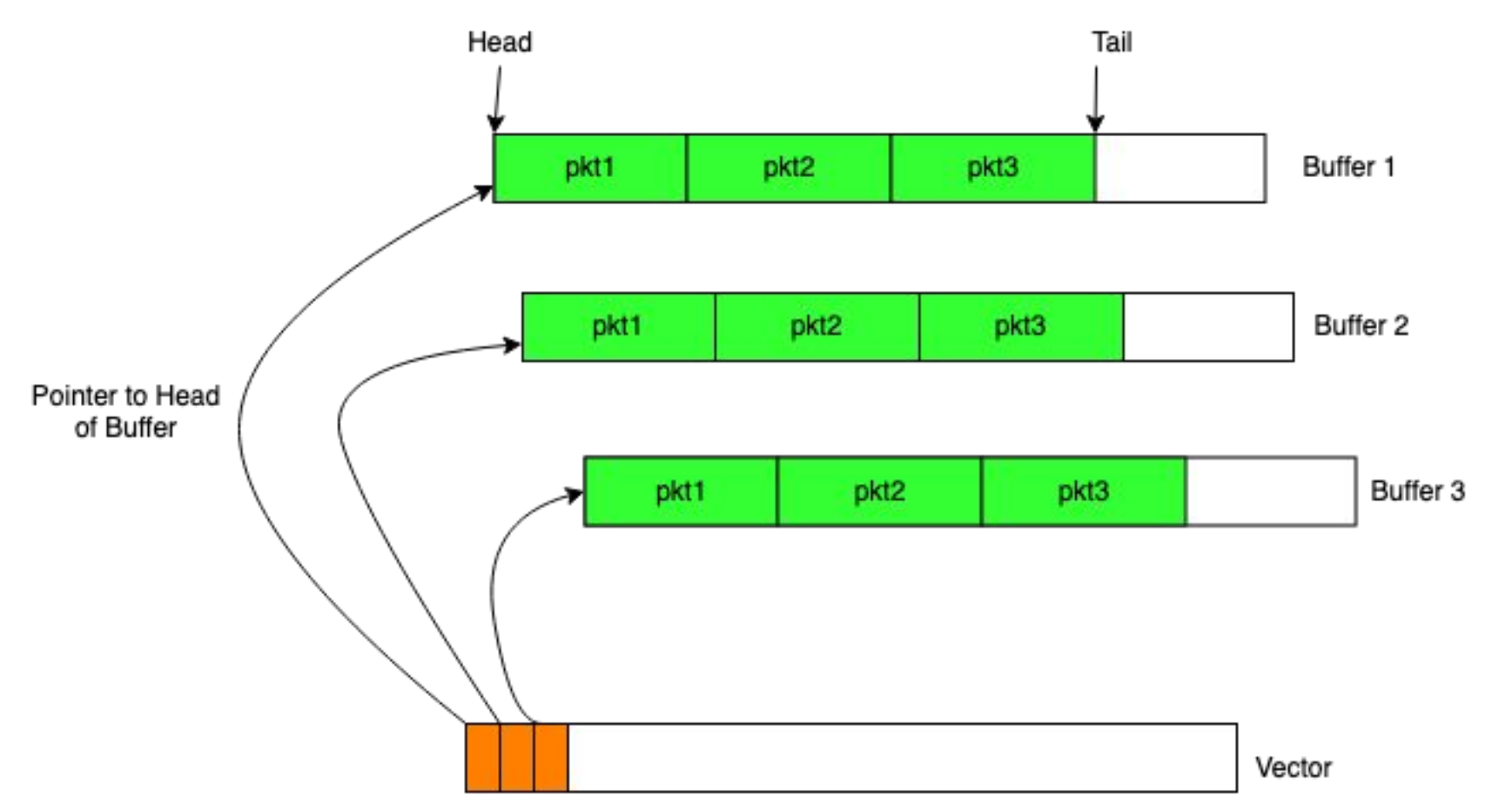
Constraints replication. FlexMock should mimic hardware constraints that may trigger bugs in the application.

Architecture



Overview of FlexMock Interface

Contiguous buffers are used to mimic SmartNIC queues that would deliver packets directly to applications that need them.



Structure of FlexBuffers used in FlexMock

FlexMock Implementation

FlexMock implementation targets Norman [1], a new OS dataplane that requires an FPGA-based SmartNIC.

Avoid runtime overheads. Buffers in a vector are prefilled with packets to avoid per packet overhead at runtime.

Debugging capabilities. The mock has optional data collection (including statistics and sent packets) to aid debugging.

Functionality test. The mock can use raw sockets to send packets to another application over the network, allowing developers to verify functionality with multiple applications.

Evaluation and Discussion

We tested FlexMock on both an echo server and Google's Maglev load balancer using 64-byte packets.

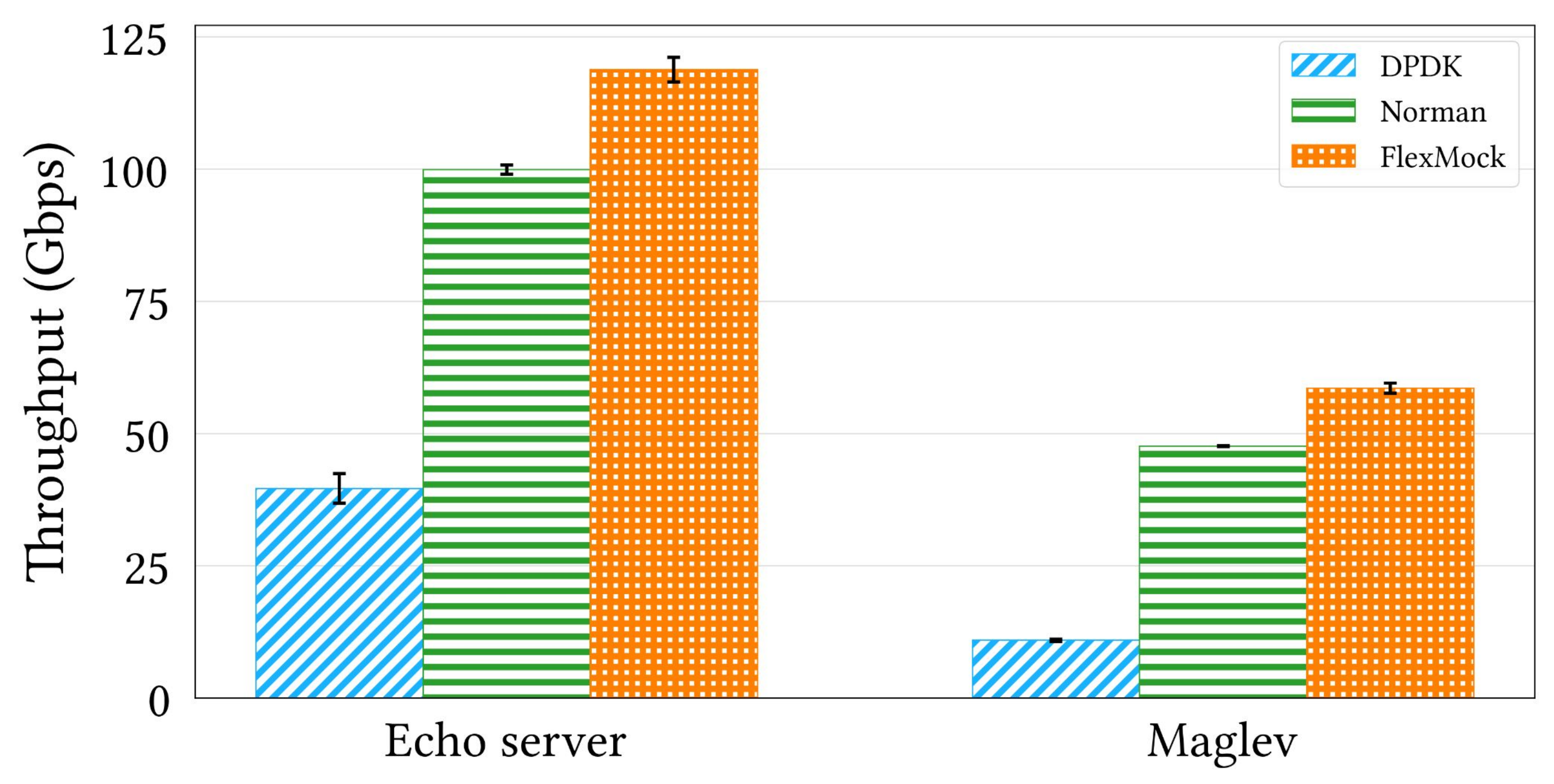
Functionality. Both applications, originally designed for Norman OS, ran normally without using the FPGA SmartNIC.

Performance. In performance mode, both applications performed better than in DPDK and Norman; this is due to avoided runtime overheads and reduction of unneeded hardware-related functionality in FlexMock.

While FlexMock does not precisely mirror the actual performance of Norman OS, it achieves the basic functionality that would be needed for developing and testing applications.

FlexMock Performance

Performance of Echo Server and Maglev using DPDK, Norman, and FlexMock (in performance mode).



References:
1. Sadok, et al. 2021. "We need kernel interposition over the network dataplane," HotOS '21