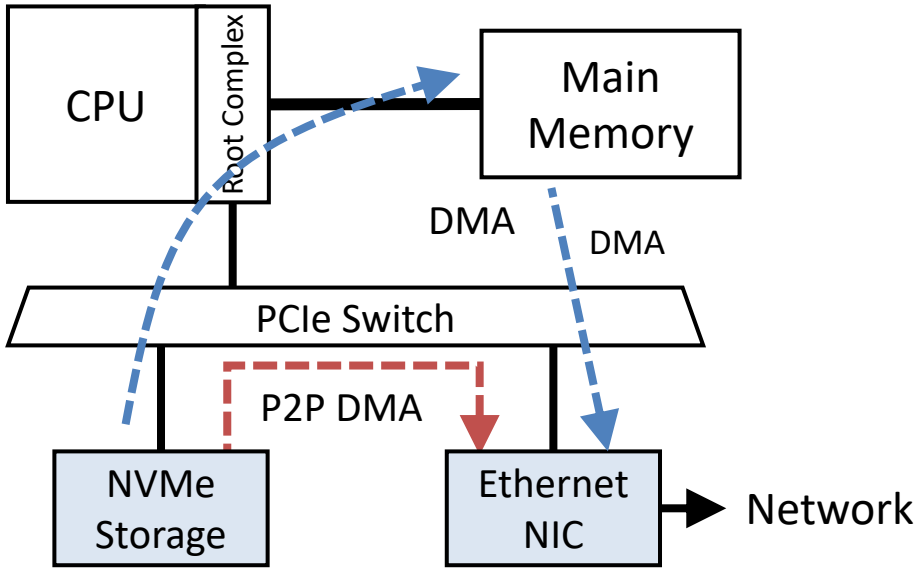
高速大容量トラフィックキャプチャ/ジェネレータの開発

中村 遼 (東京大学)

- P2P DMAによる大容量トラフィックキャプチャ/ジェネレータ
  - Ethernet NICとNVMe SSDで直接データをやりとりする
  - メインメモリとCPUをバイパスすることでデバイス性能の限界に挑む
- P2P DMAを行うためには
  - デバイス上のメモリをDMAの宛先にする
  - しかし、既存のデバイスドライバは I/O用のバッファの位置を指定できない
    - 基本的にメインメモリから割り当て



例: Linuxにおけるパケットバッファの割り当て

```
struct sk_buff *__alloc_skb(unsigned int size, gfp_t gfp_mask,
                            int flags, int node)
```

実装

- ユーザ空間からデバイス上のメモリを制御するライブラリを実装
  - デバイス上のメモリをカーネル空間でalloc、ユーザ空間へmmap()
  - ユーザ空間のアプリケーションからデバイス上のメモリを透過的に利用することができる
- 本ライブラリを2つのカーネルバイパス型のドライバに統合
  - Ethernet NIC: netmap, https://github.com/luigirizzo/netmap
  - NVMe: UNVMe, https://github.com/MicronSSD/unvme
  - この2つを組み合わせて、 P2P DMA可能なトラフィックキャンプチャ/ジェネレータを実装

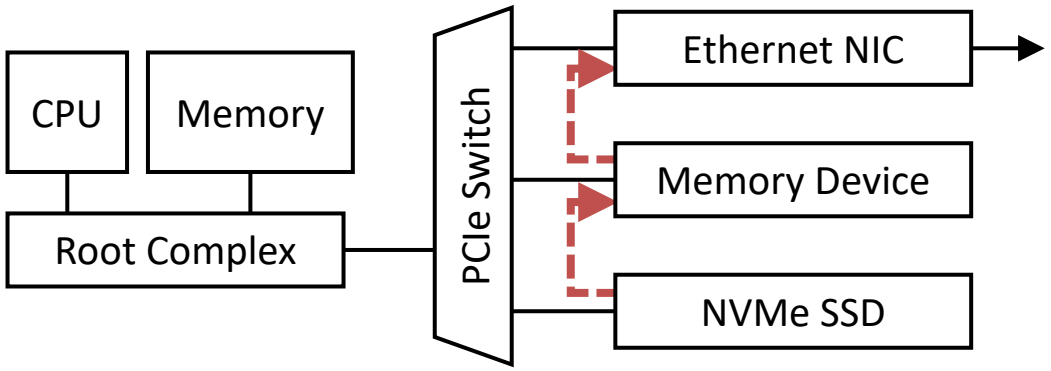
実装したライブラリのAPI

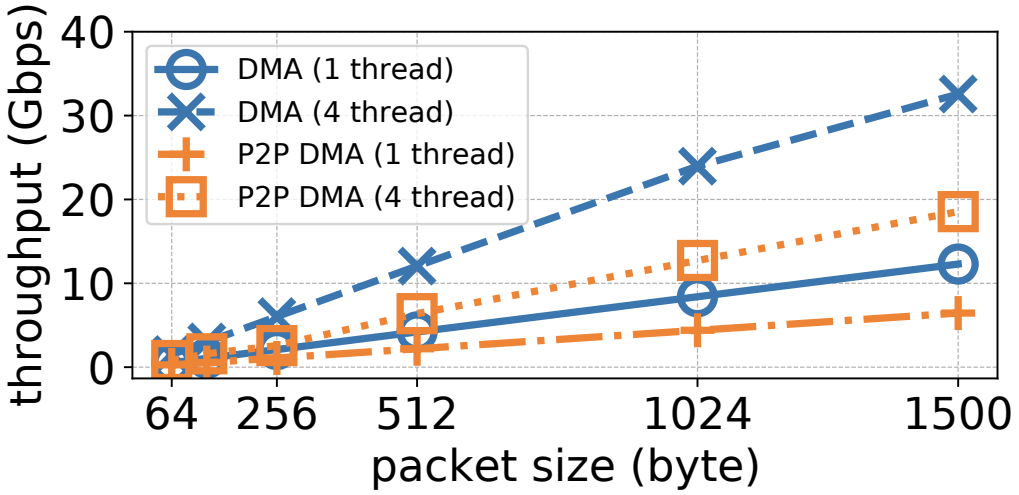
```
pop_mem_t *pmem;
pop_buf_t *pbuf;
void *data;

pmem = pop_mem_init("17:00.0", 4096);
pbuf = pop_buf_alloc(pmem, 2048);
data = pop_buf_data(pbuf);
```

実験

- 40Gbps NIC + NVMeストレージで性能計測
  - 2019年現在CMB対応のNVMe SSDは市販されていないため、FPGAベースのメモリデバイスをEthernet NICとNVMeの間に挟むことで、CPUとメインメモリを通らない構成を構築し、性能を計測
  - 結果、この構成が原因となって、P2P DMAでは性能を発揮することができなかった





まとめと今後の課題

- P2P DMAを行うためのユーザランドライブラリの実装
  - netmap (Ethernet NIC)とUNVMe (NVMe)に本ライブラリを統合
  - P2P DMAを用いるトラフィックジェネレータ/キャプチャを実装
- 性能計測
  - 現時点ではEthernet NICとNVMe SSDで直接P2P DMAはできない
  - そのためFPGAベースのメモリデバイスを中間バッファとして用いたがこれが原因となって性能を引き出すことはできなかった
- MMIOでアクセス可能なメモリを持つデバイスの必要性
  - NVMeではController Memory Buffer機能がそれにあたるが、現時点で市販されている対応製品は無し
  - 今後FPGAやSmart NICなどの利用を検討