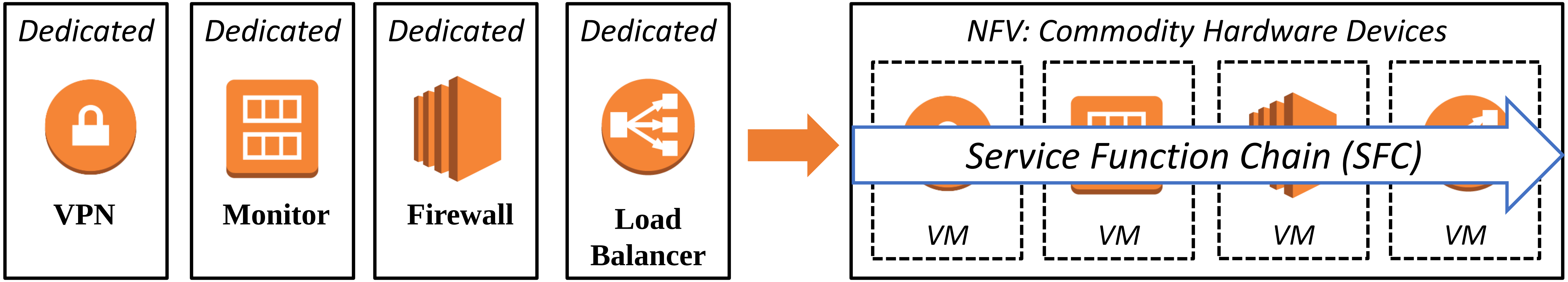
GEN: A GPU-Accelerated Elastic Framework for NFV

Zhilong Zheng Jun Bi Chen Sun Heng Yu Hongxin Hu
Zili Meng Shuhe Wang Kai Gao Jianping Wu

Tsinghua University
Clemson University

Network Function Virtualization (NFV)



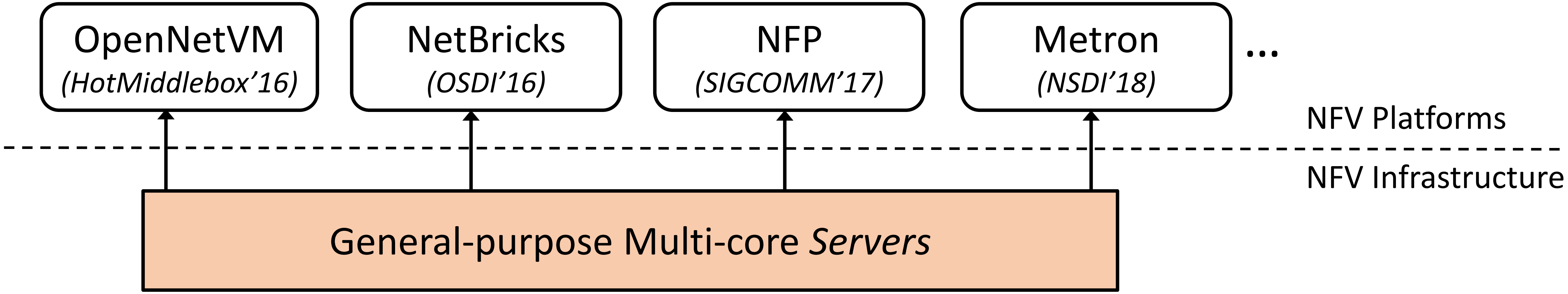
Virtualization Techniques

Low cost

Elasticity control

Service provisioning flexibility

CPU-based NFV



- Problems
  - Low *performance* with negative improvement expectation
  - Coarse-grained *scaling*

Problems of CPU-based NFV

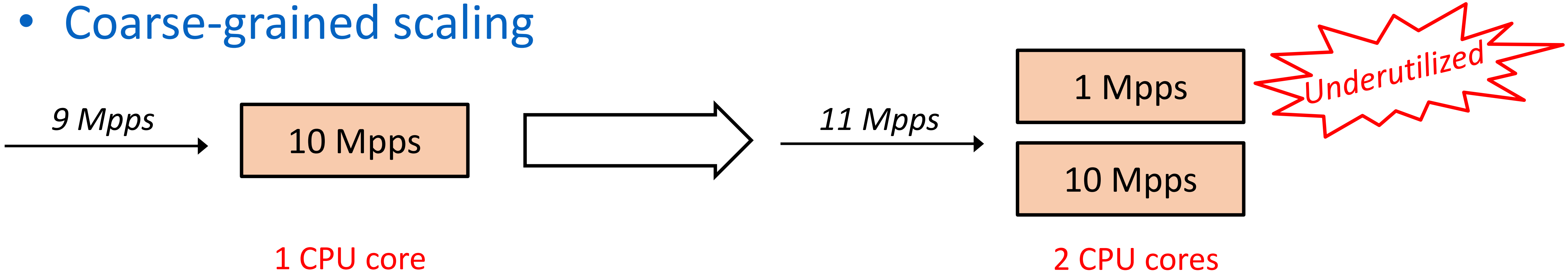
- Low performance with negative improvement expectation
  - Hard to achieve high performance (e.g., 40~100Gbps) for a wide range of NFs

IPSec *(AES & SHA1)*	NIDS (Aho-Corasick)
2.6 ~ 7.7 Gbps	4.2 ~ 10.4 Gbps

E5-2650 v2 (8 Cores, 2.6 GHz)

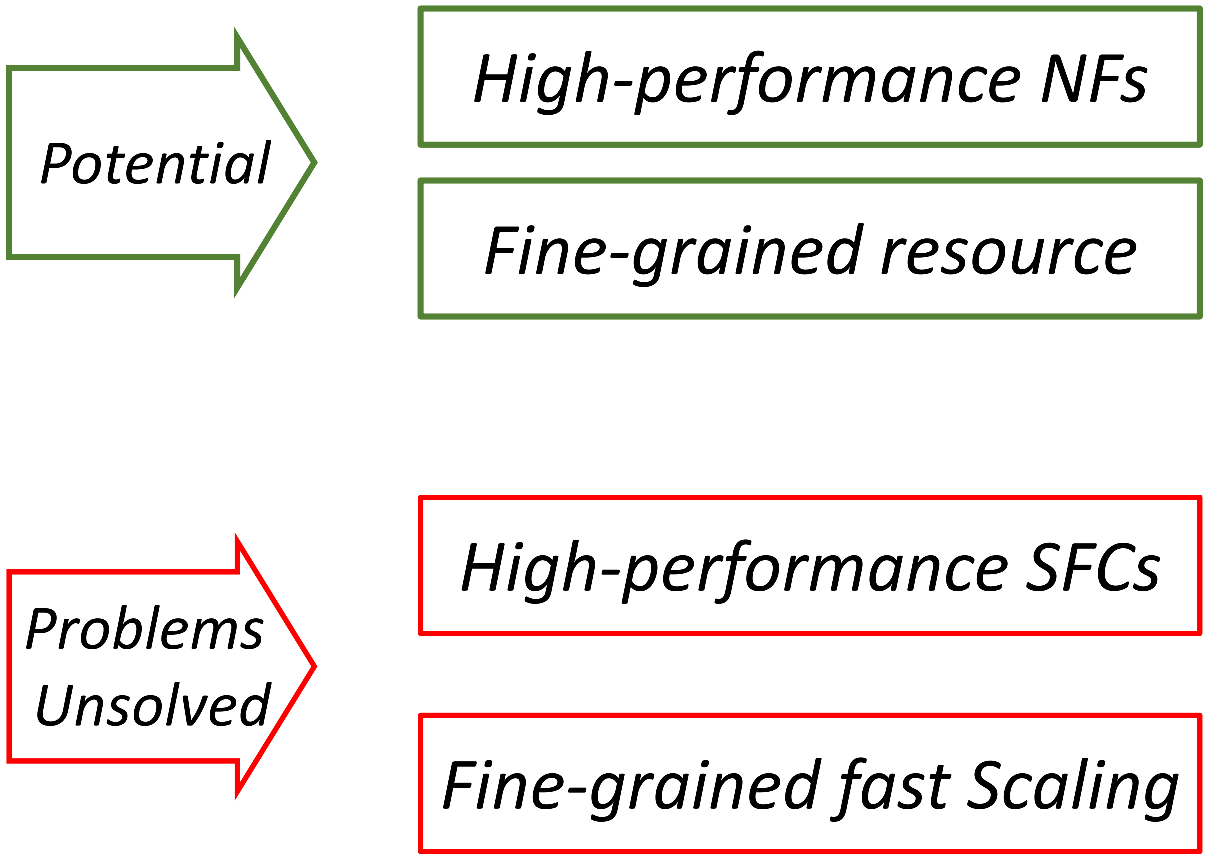
Go, Younghwan, et al. "APUNet: Revitalizing GPU as Packet Processing Accelerator." *NSDI*. 2017.

  - The slow/end of Moore's Law
- Coarse-grained scaling



GPU as An Accelerator for NFV

- Benefits of GPU
  - Massive processing cores
  - Fine-grained computing units
- Existing work
  - Router *(PacketShader, SIGCOMM'10)*
  - SSL proxy *(SSLShader, NSDI'11)*
  - NIDS *(Kargus, CCS'12)*
  - IPSec *(NBA, EuroSys'15)*
  - NFV framework *(G-NET, NSDI'18)*

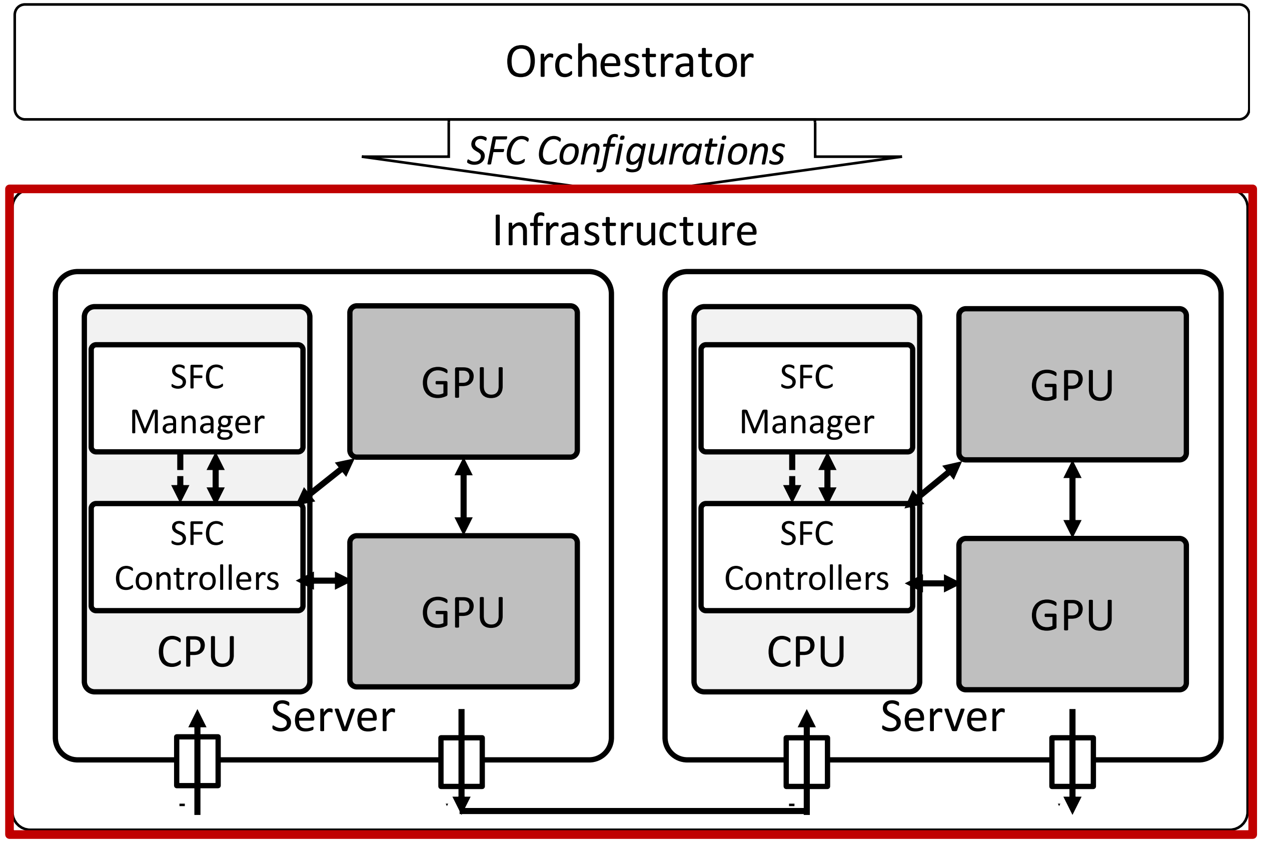


GEN exploits ***GPU*** *to support*

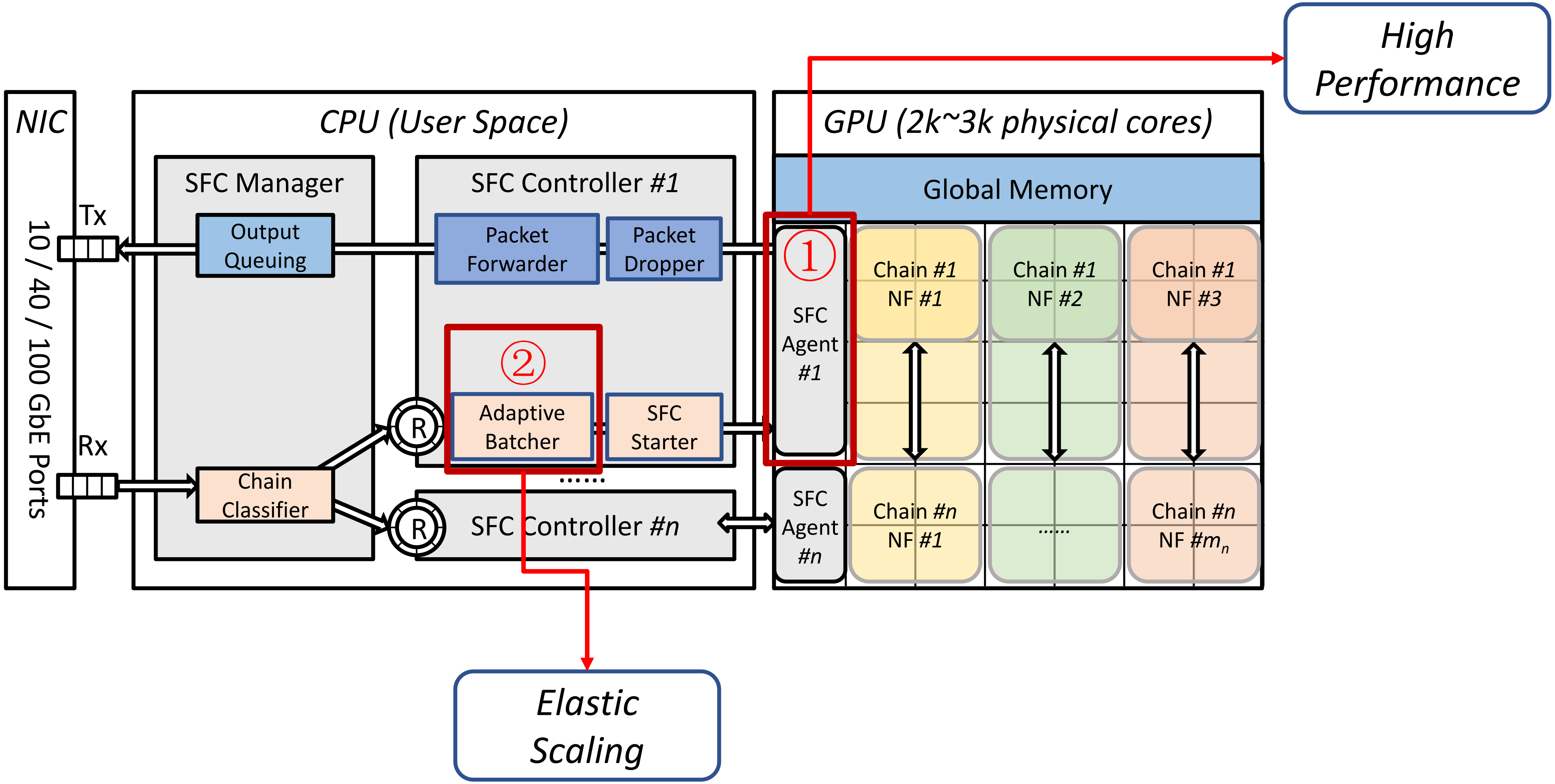
high-performance SFCs

with ***fine-grained scaling***

GEN Framework Overview

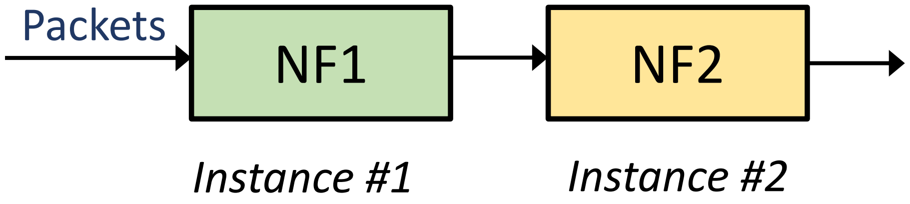


Infrastructure Design

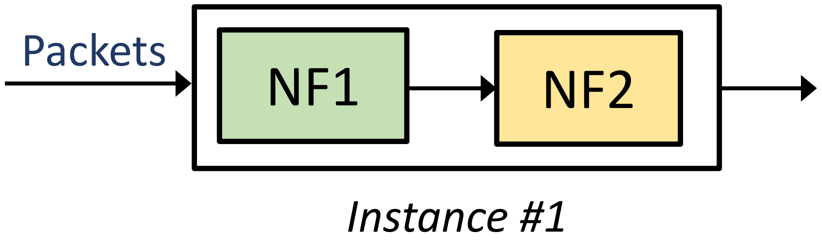


Problem #1: SFC Model Selection

Pipelining



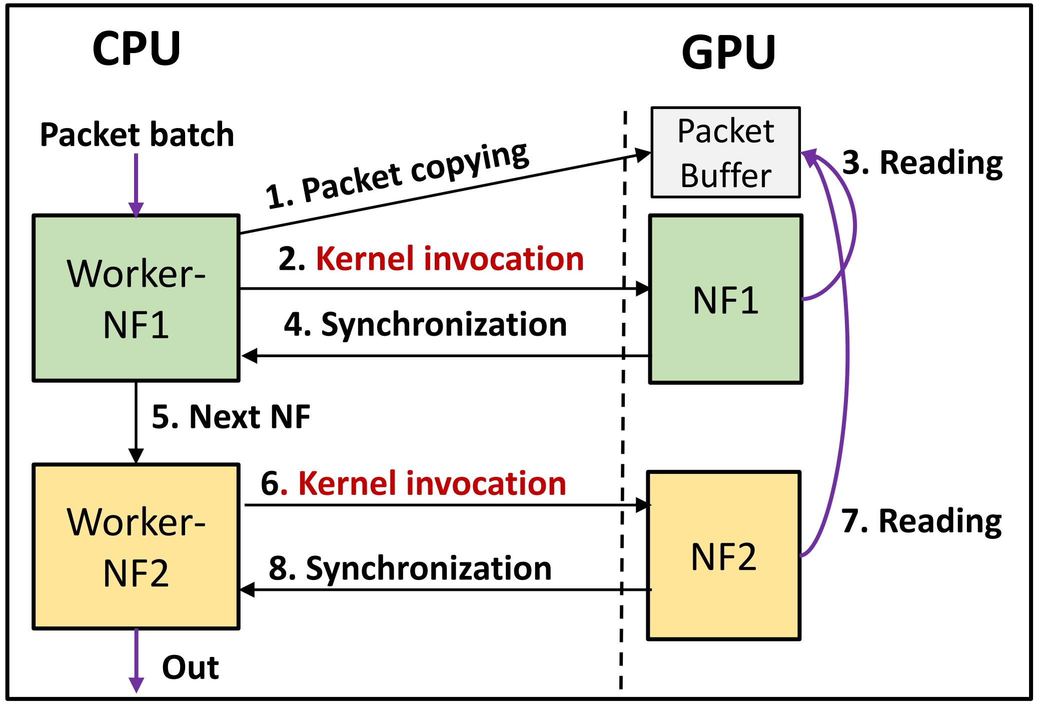
Run-to-completion (RTC)



SFC Model Selection: Pipelining

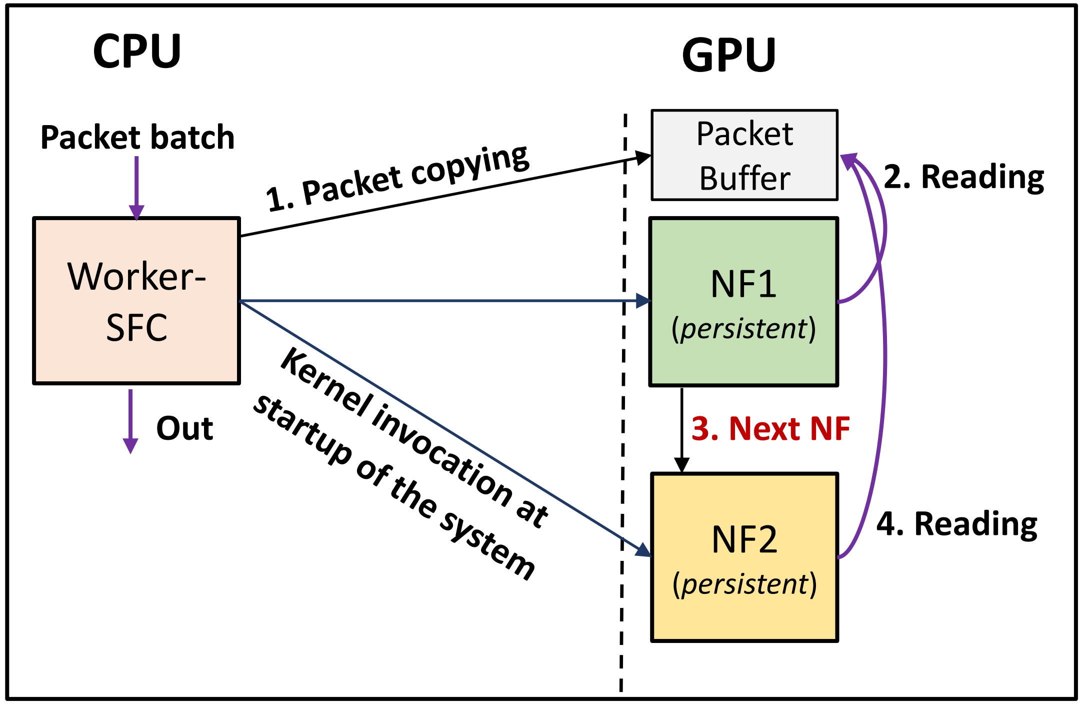
- Two potential ways to support pipelining in GPU

Sequenced invocations



High overhead from frequent kernel invocations (~5us per invocation)

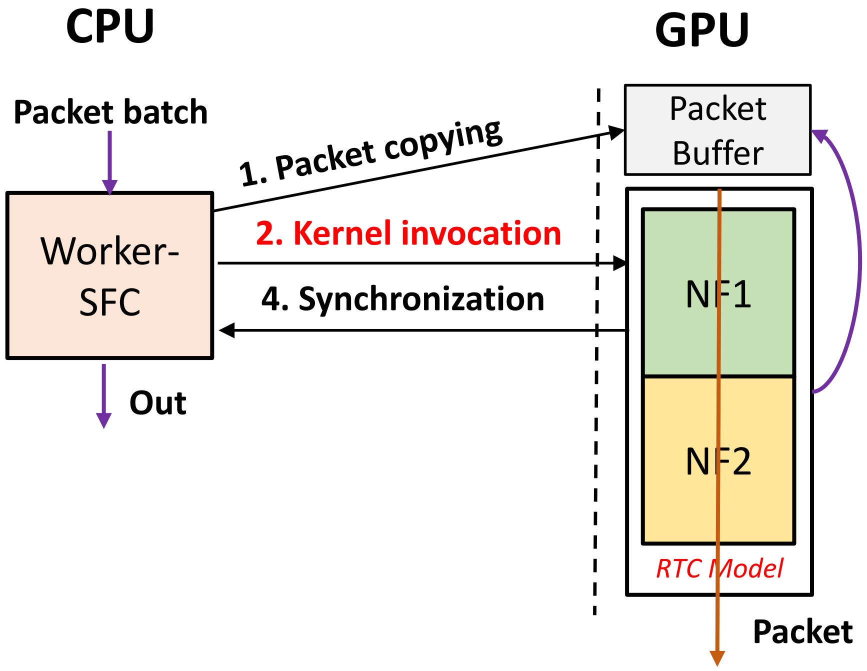
Persistent kernels



Hard and costly scaling

SFC Model Selection: RTC

- RTC-based Model



Less kernel invocations (once per SFC)

Easier scaling (not persistent)

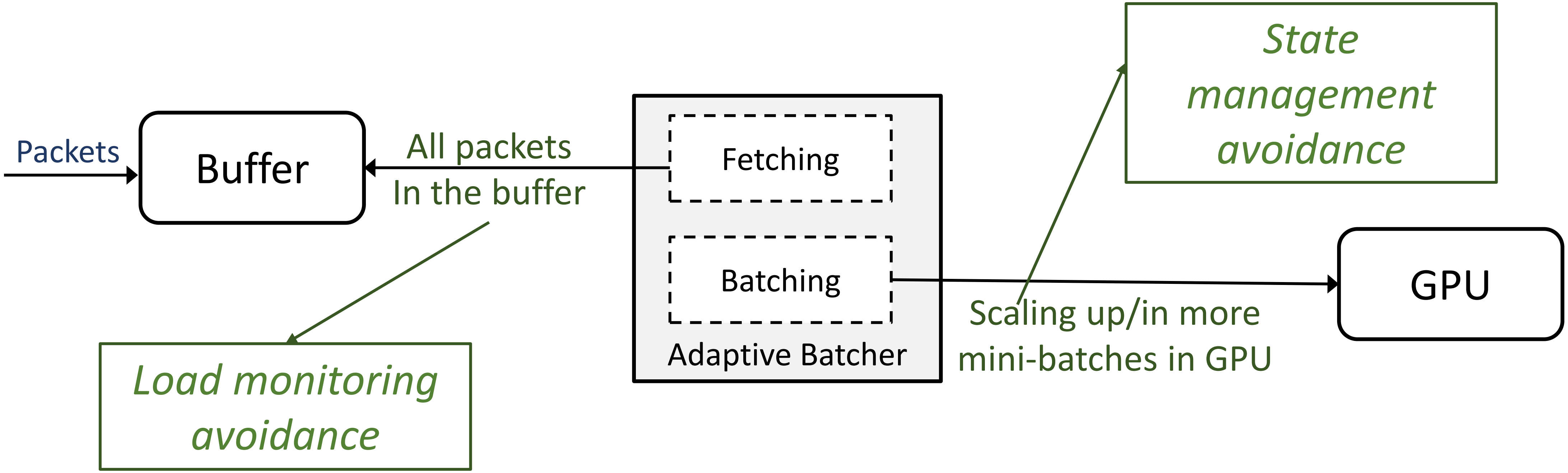
- NFs are integrated into a specific *SFC Agent* ⟸ *kernel fusion*
- SFC Agent (in GPU) is Launched by *SFC Starter (in CPU)*

Problem #2: Elastic Scaling

- Avoid monitoring NF load for scaling
  - Avoid deciding *when* to scale
  - Avoid deciding to what *extent* an NF should be scaled
  - Avoid considering how to quickly carry out NF scaling
- Avoid state management caused by scale out / in
  - Intuition: Use scale up / down to avoid state management
- *Adaptive Batcher*

Elastic Scaling – Adaptive Batcher

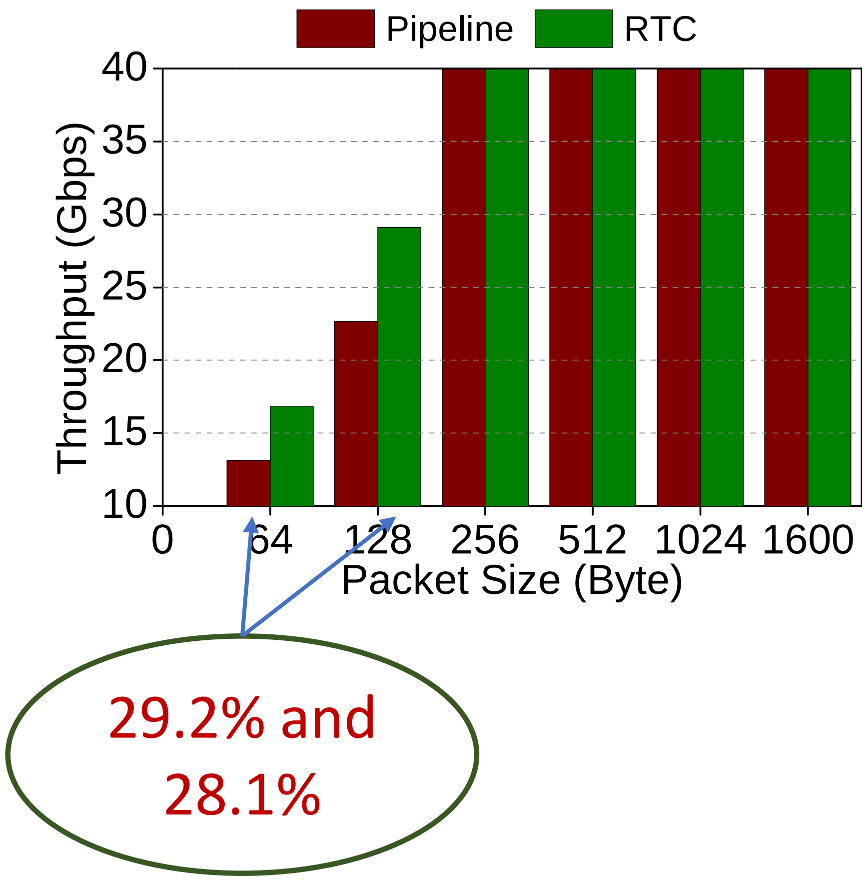
- Design of the adaptive batcher
  - Keeping the buffer occupancy at a low level
  - Scaling up/in GPU resource provisioning

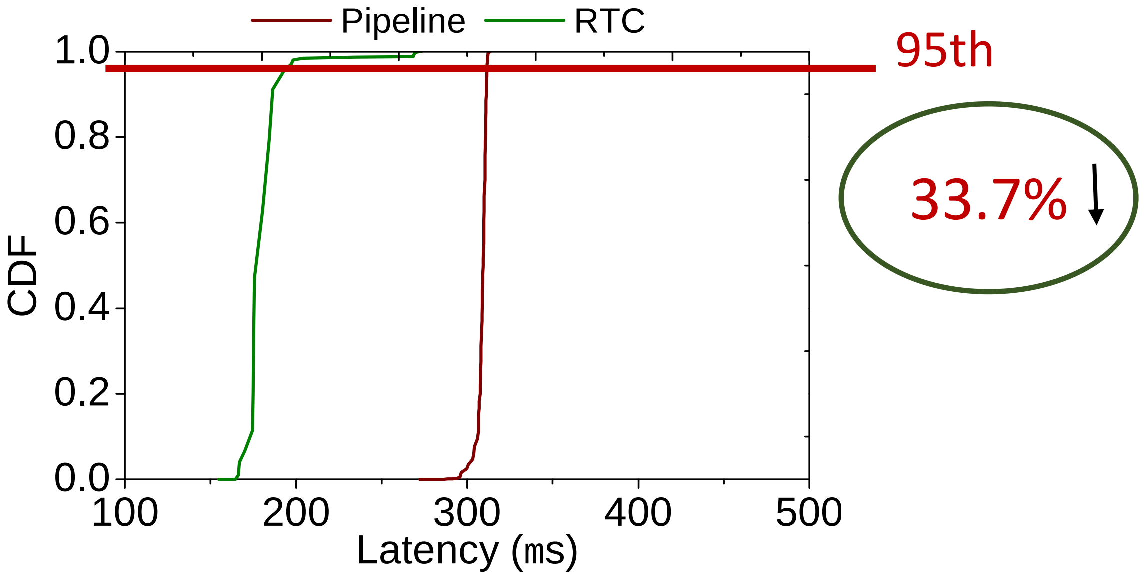


Preliminary Evaluation

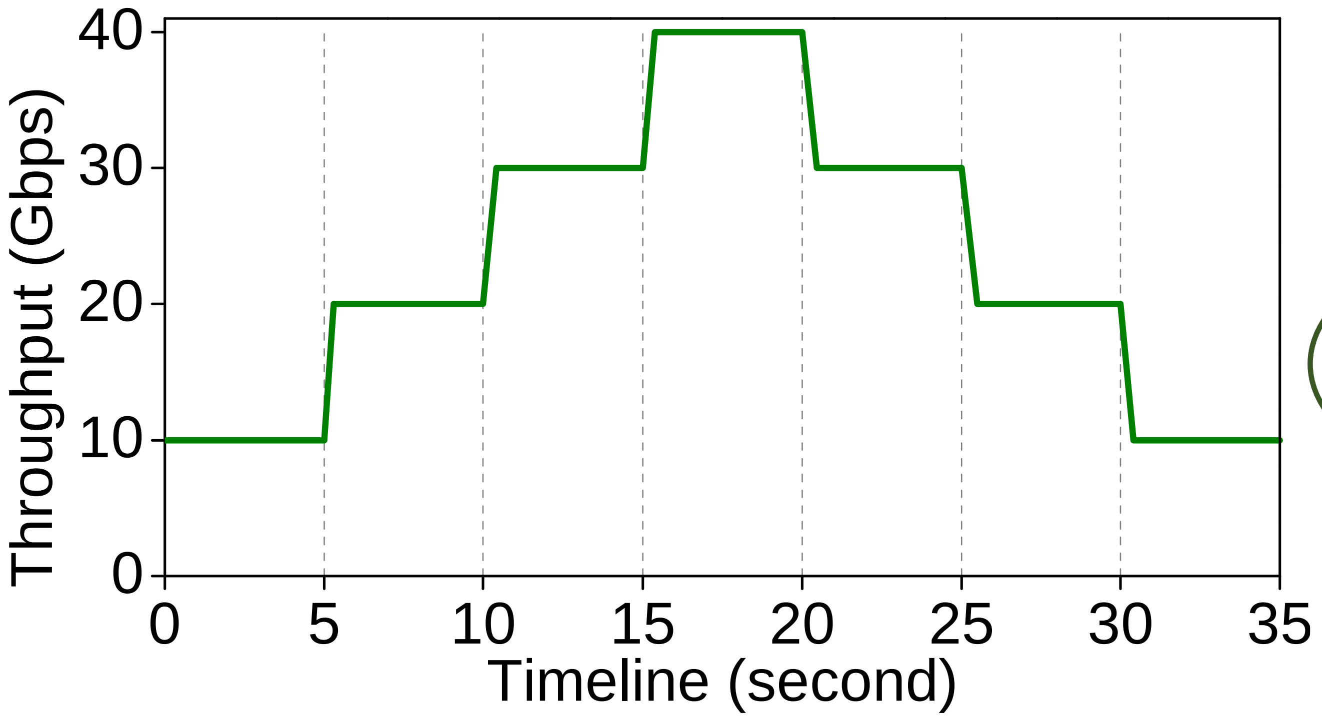
- Hardware
  - CPU: Two Intel Xeon E5-2650 v4 (10 physical cores)
  - GPU: NVIDIA TITAN Xp
  - NIC: Two Intel X520 (40 Gbps in total)
- Software
  - DPDK 17.11 for networking IO
  - CUDA 8.0 for GPU programming
- NFs & SFCs
  - IPV4Router (1k entries) → NIDS (3k rules) → IPSec (SHA1 & AES-128-CBC)

Performance of RTC *vs.* Pipelining





Fast Elastic Scaling



Fast converging (< *100ms*)

Conclusion and Future Work

- Gen: a GPU-accelerated elastic framework for NFV
  - High-performance SFC
  - Elastic scaling

- Future work
  - More SFC performance enhancement in GPU
  - Coordination between CPU and GPU
  - Impact of dynamic traffic load

Thank You

http://netarchlab.tsinghua.edu.cn