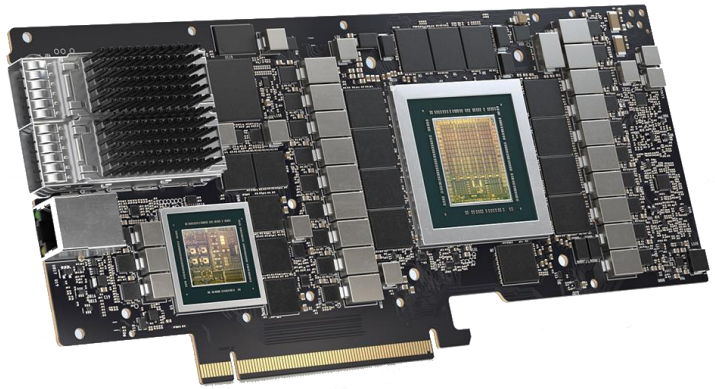
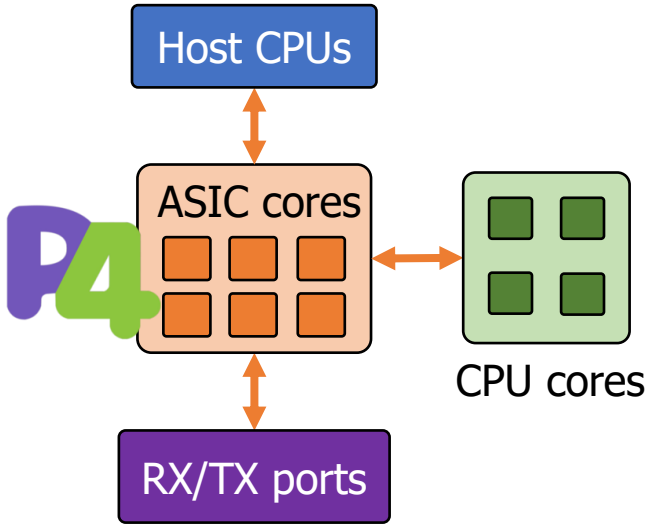
Unleashing SmartNIC Packet Processing Performance in P4

Jiarong Xing, Yiming Qiu, Kuo-Feng Hsu, Songyuan Sui,
Khalid Manaa, Omer Shabtai, Yonatan Piasetzky, Matty Kadosh,
Arvind Krishnamurthy, T. S. Eugene Ng, Ang Chen

Background: Multicore-based SmartNICs

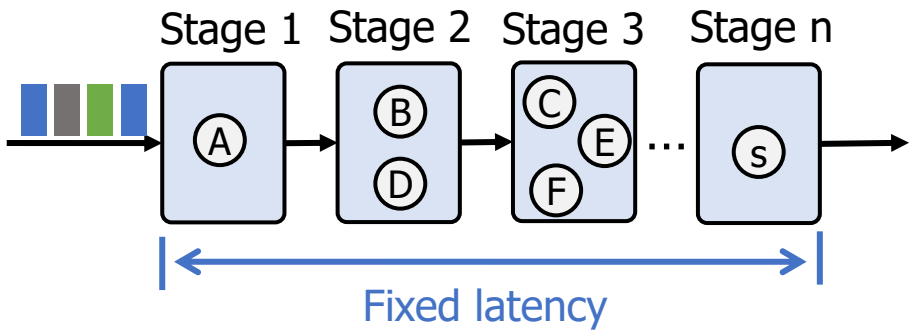




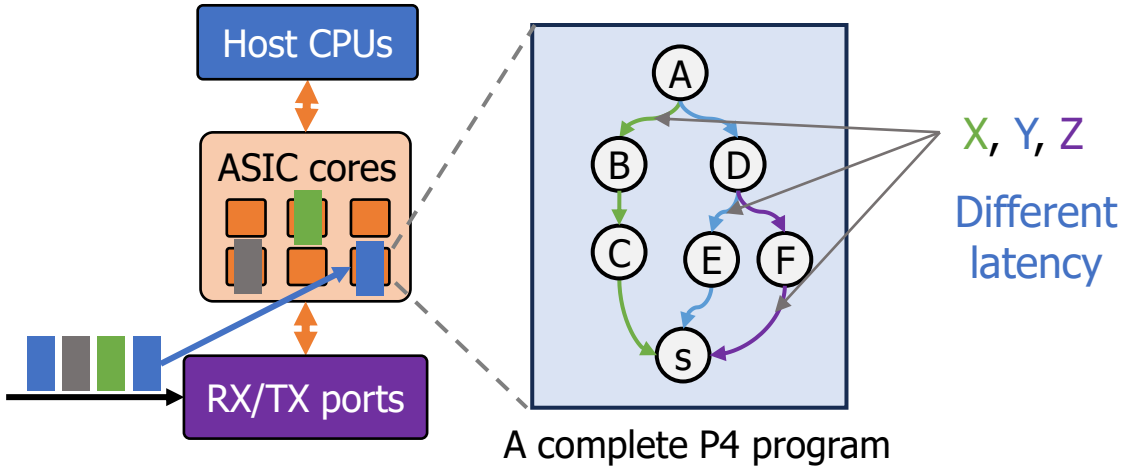
Example: Nvidia BlueField

- Widely deployed infrastructure for high-performance data centers
  - Many networking functions are offloaded onto SmartNICs for acceleration
- Our focus: Multicore-based SmartNICs
- P4 is emerging as a unified SmartNIC programming language

Problem: Lack of performance optimization



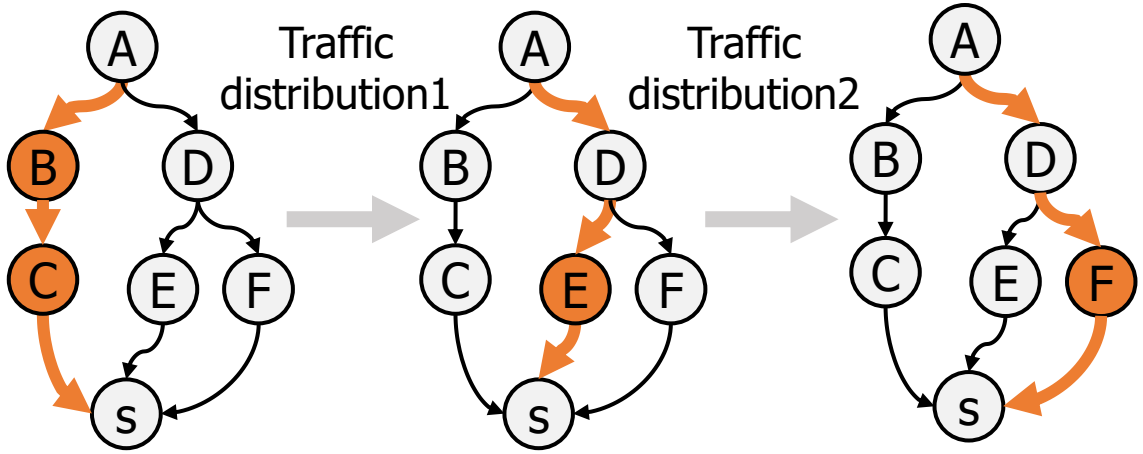
Stage-based pipeline processing
Deterministic performance



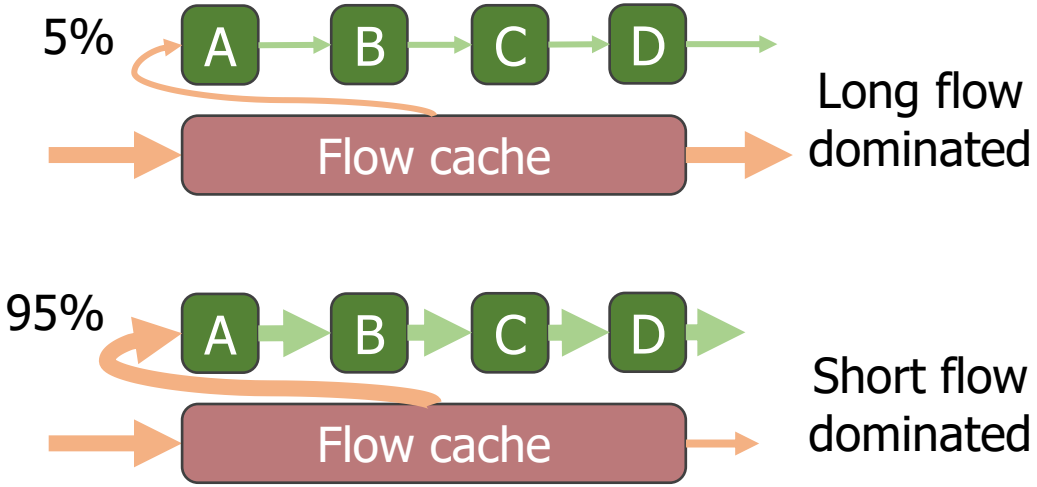
Run-to-completion (RTC) processing
High variability in performance

- Multicore SmartNICs are flexible but lack performance guarantee
  - RTC has varied performance depending on workloads and implementations
  - Given a workload, inefficient implementations lead to low performance
- Performance optimization is necessary and imperative
- Today's P4 compilers optimize for resource not performance

Challenge: Dynamic profile changes



Program bottleneck change



Cache hit rate change

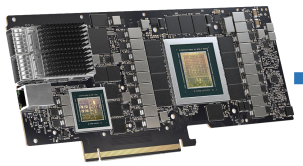
- Multicore SmartNIC performance varies based on runtime profiles
  - e.g., traffic distributions, flow lengths, table entries, etc.
  - Unknown at compile time and dynamically change at runtime
- Compile time optimization is hard to produce desired performance

Solution: Runtime profile-guided optimization

Chameleon

Change color to match

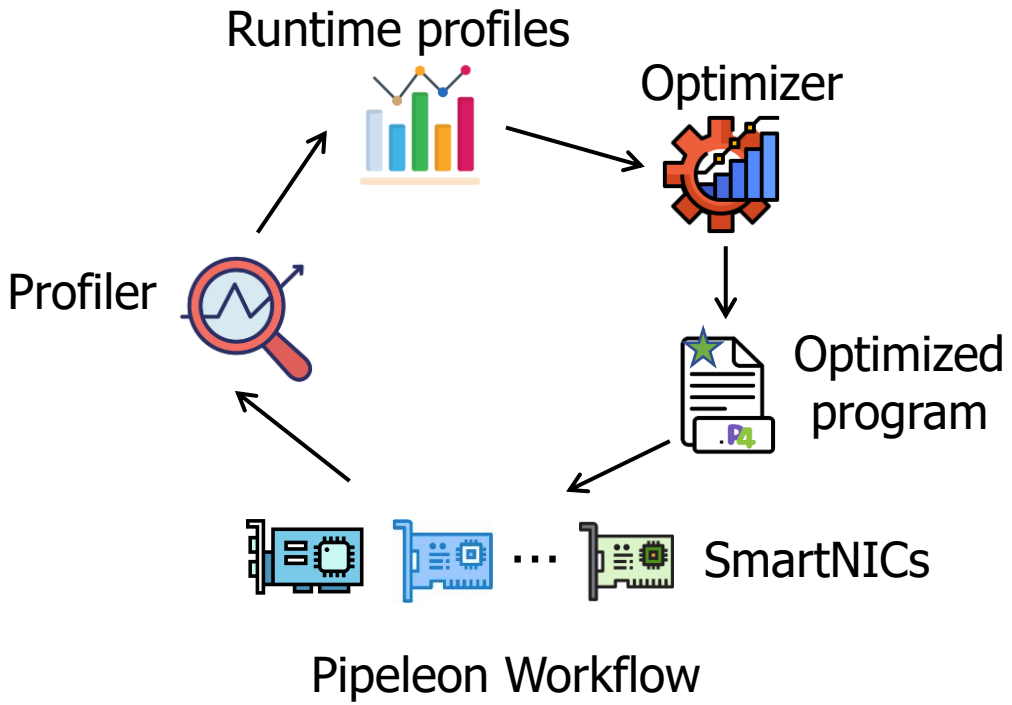
Surroundings



Pipeleon

Change implementation to match

Runtime profiles



- Dynamic adaptation enabler: Runtime programmable ASICs
  - Our prior work FlexCore at NSDI'22 & HotChips'23
  - In-place live update with no downtime
- Key question: How and where to optimize SmartNIC P4 programs?

P4 performance optimizations in Pipeleon

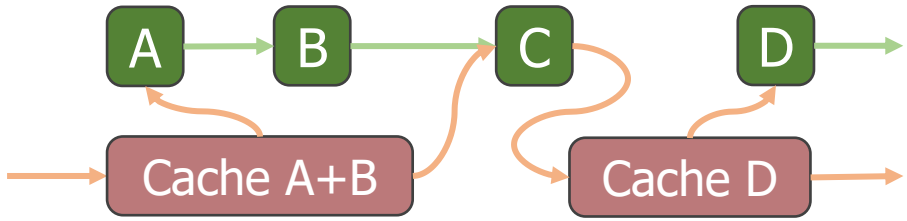


Table caching

Skip complex table matches using flow caches

Support multiple smaller caches

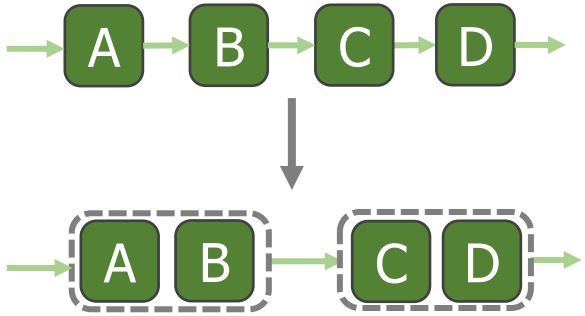


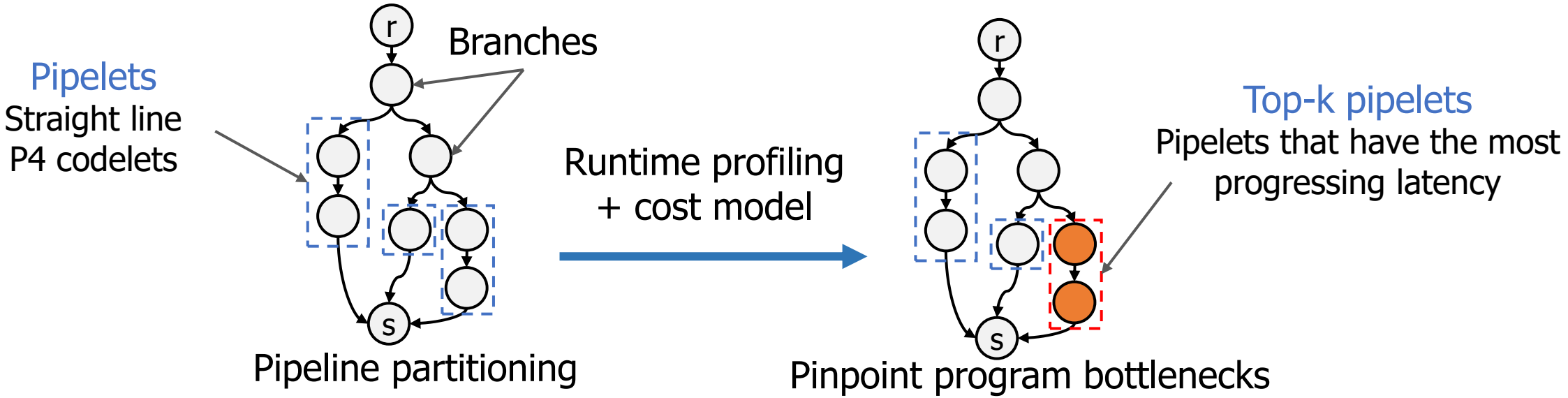
Table merging

Perform multiple tables together to reduce memory accesses

Take table types into consideration

- A set of performance-oriented P4 optimizations
- Customized for multicore-based SmartNIC
- See more optimizations in our paper

Top-k pipelet based optimization



- How to maximize the performance gain within the resource constraints?
  - Optimizations produce different gains and incur resource overheads
- Exhaustive search over the whole program has a huge large space
- Solution: Top-k pipelet based optimization
  - Insight: Optimizing bottlenecks tends to generate more performance gain

Evaluation highlights

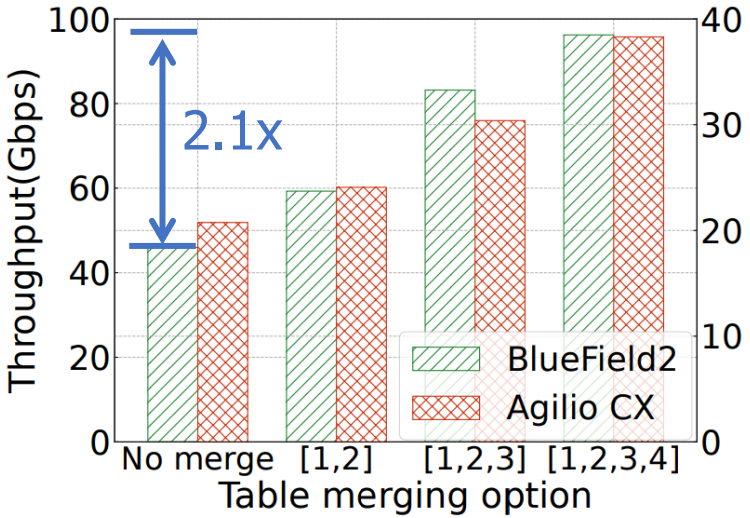
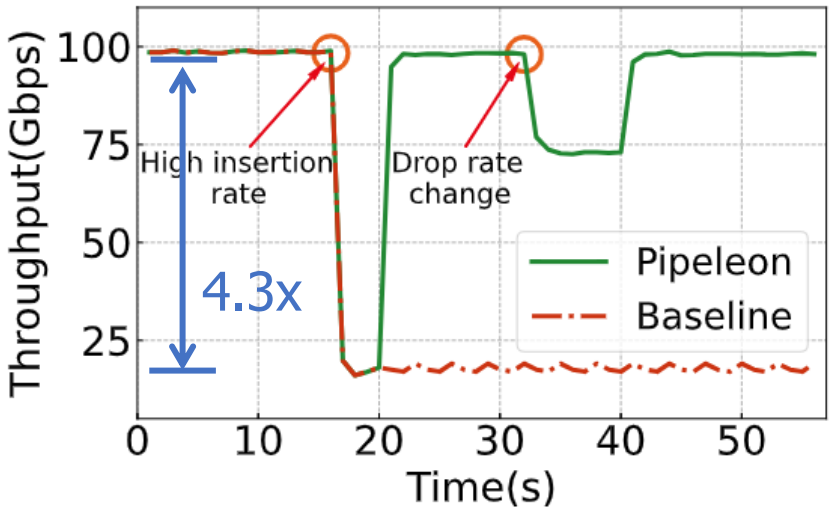
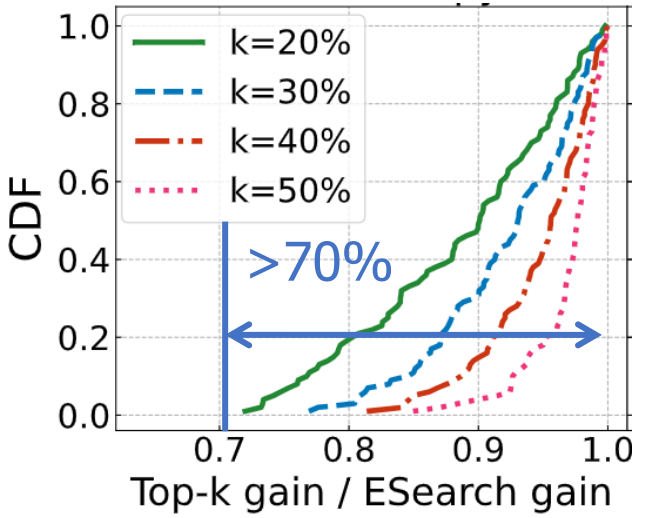


Table merging benefits



Service LB on BlueField2



Top-k search effectiveness

- Table merging improves throughput by up to 2.1x on BlueField2 and Netronome
- Pipeleon can quickly adapt to runtime profile changes
- Top-20% pipelet search can achieve >70% of performance gain achieved by exhaustive search while reducing the search time by 8.2x

Summary

- P4 compilers lack performance optimization for multicore SmartNICs
- Pipeleon: An automated performance optimization framework
  - A set of P4 optimization techniques
  - Runtime adaptation based on profiling
- Our prior work on runtime programmable infrastructures
  - Resource Transmutable Network Processing ASIC, HotChips'23
  - Runtime Programmable Switches, NSDI'22
  - A Vision for Runtime Programmable Networks, HotNets'21
  - See more details here: https://jxing.me/#pub

Code

My webpage