Whippersnapper: A P4 Language Benchmark Suite

Huynh Tu Dang, Han Wang, Theo Jepsen, Gordon Brebner, Changhoon Kim, Jennifer Rexford, Robert Soulé, Hakim Weatherspoon

The Rise of P4

Enabling to program reconfigurable switch chips

Compilers:

- P4c, PISCES, P4FPGA, Xilinx SDNet, Barefoot Tofino, etc.

Targets:

- CPUs, NPUs, FPGAs, and ASICs

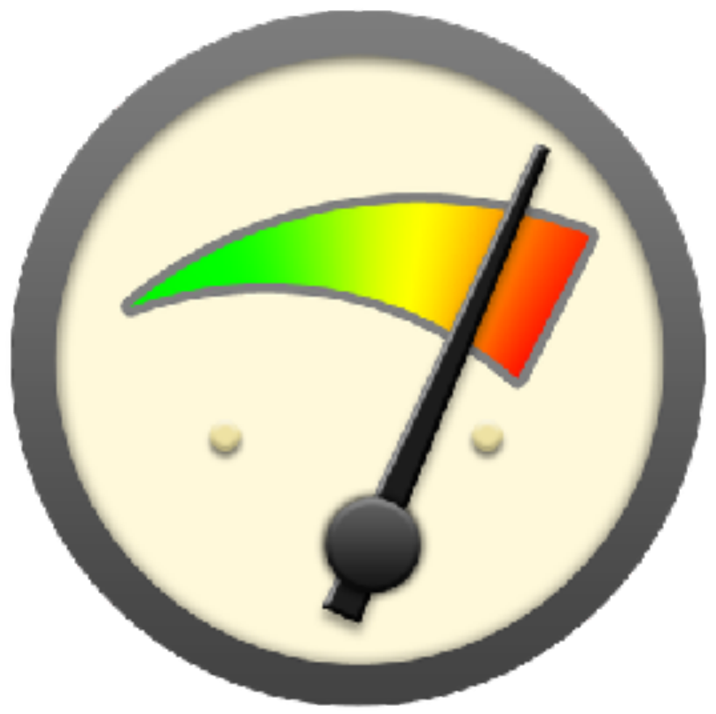
Is My P4 Compiler Competitive?

Performance is important to achieve a competitive advantage

P4 lacks a tool to evaluate performance

As P4 and tools move beyond immaturity,

A P4 benchmark is in high demand



P4 Benchmark Challenges

P4 is a common API for diverse target platforms

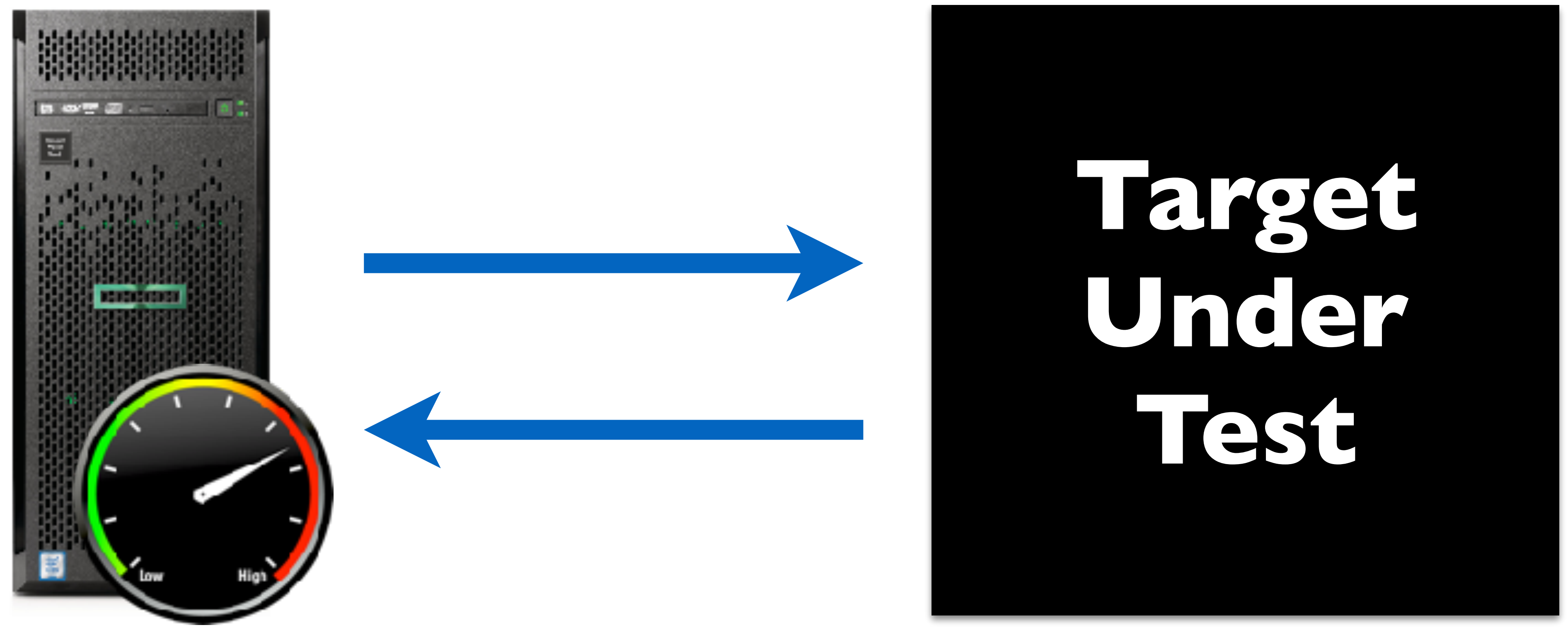
- Metrics on one target may not be relevant on another
- Collecting target-specific metrics is difficult

What are representative applications and workloads for a P4 benchmark?

Whippersnapper:
P4 Benchmark Suite

Copes with target heterogeneity

- Platform-Independent benchmark
- Platform-Specific benchmark
- Black-box benchmarking methodology



Synthetic benchmark based on core language features

Platform-Independent Benchmark

Feature	Parameter
Parsing	#Packet headers #Packet fields #Branches in parse graph
Processing	#Tables (no dependencies) Depth of pipeline Checksum on / off
State Accesses	#Writes to same/different registers #Reads to same/different registers
Packet Modification	#Header adds #Header removes

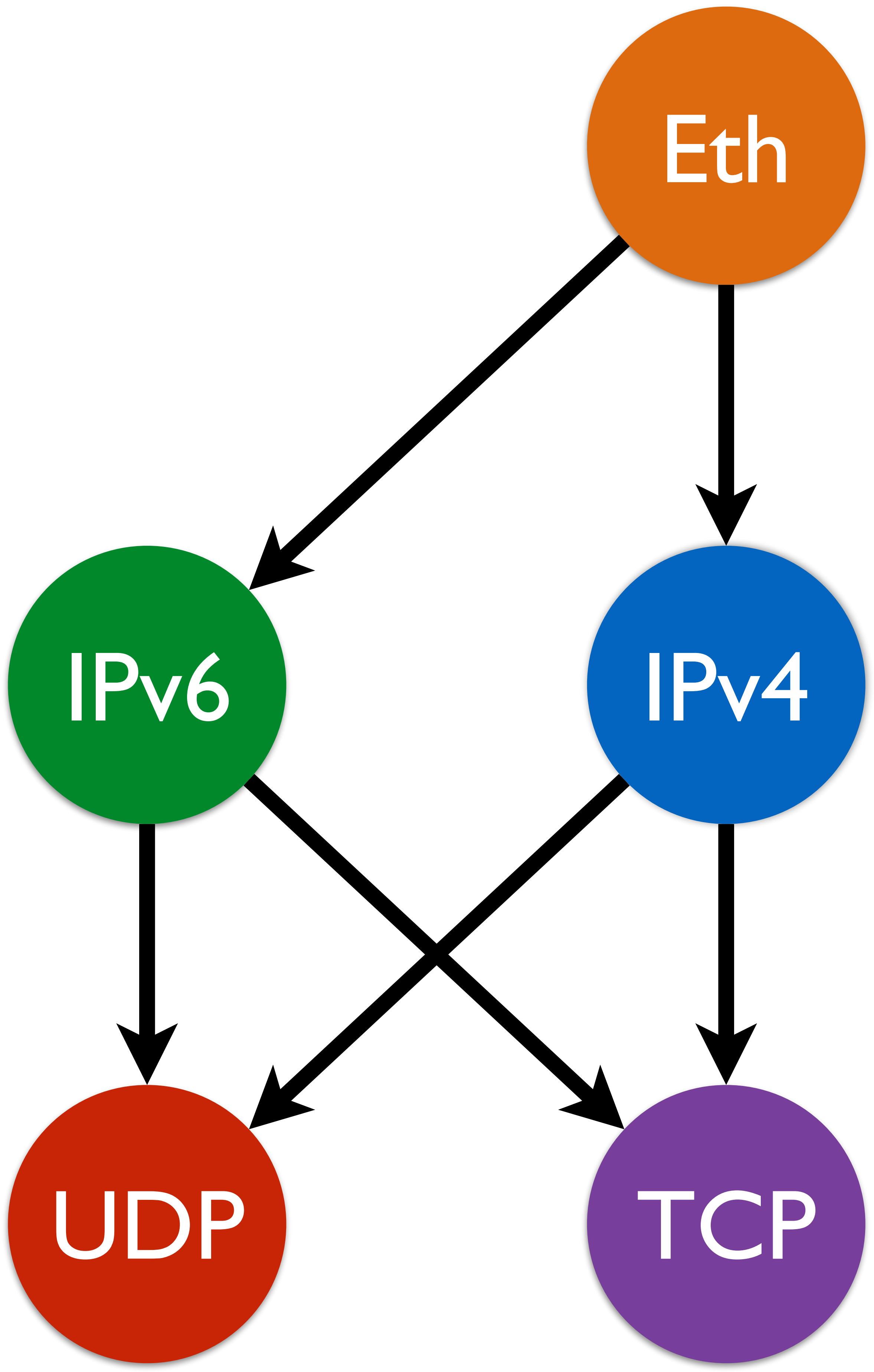
Parsing

Parsers are often implemented as State Machine

Vertices are Parse States and **Edges** are Transitions

Parameters:

- Number of packet headers
- Number of header fields
- Number of parser transitions



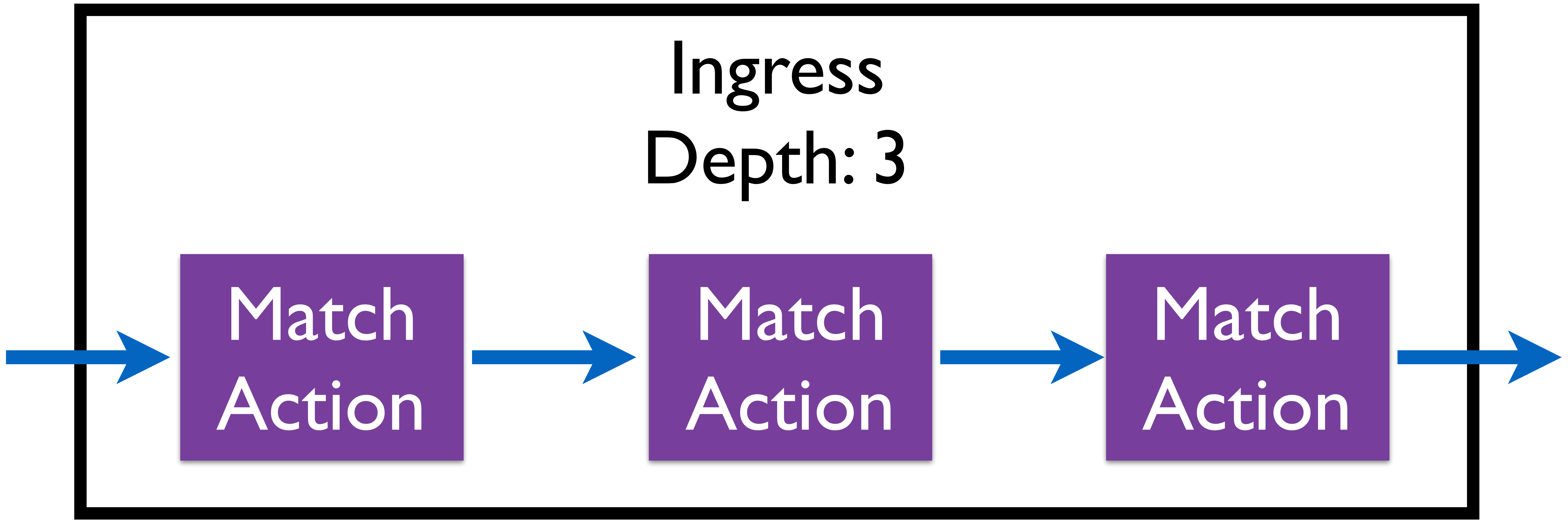
Processing

Match-Action tables placed sequentially in an ingress pipeline

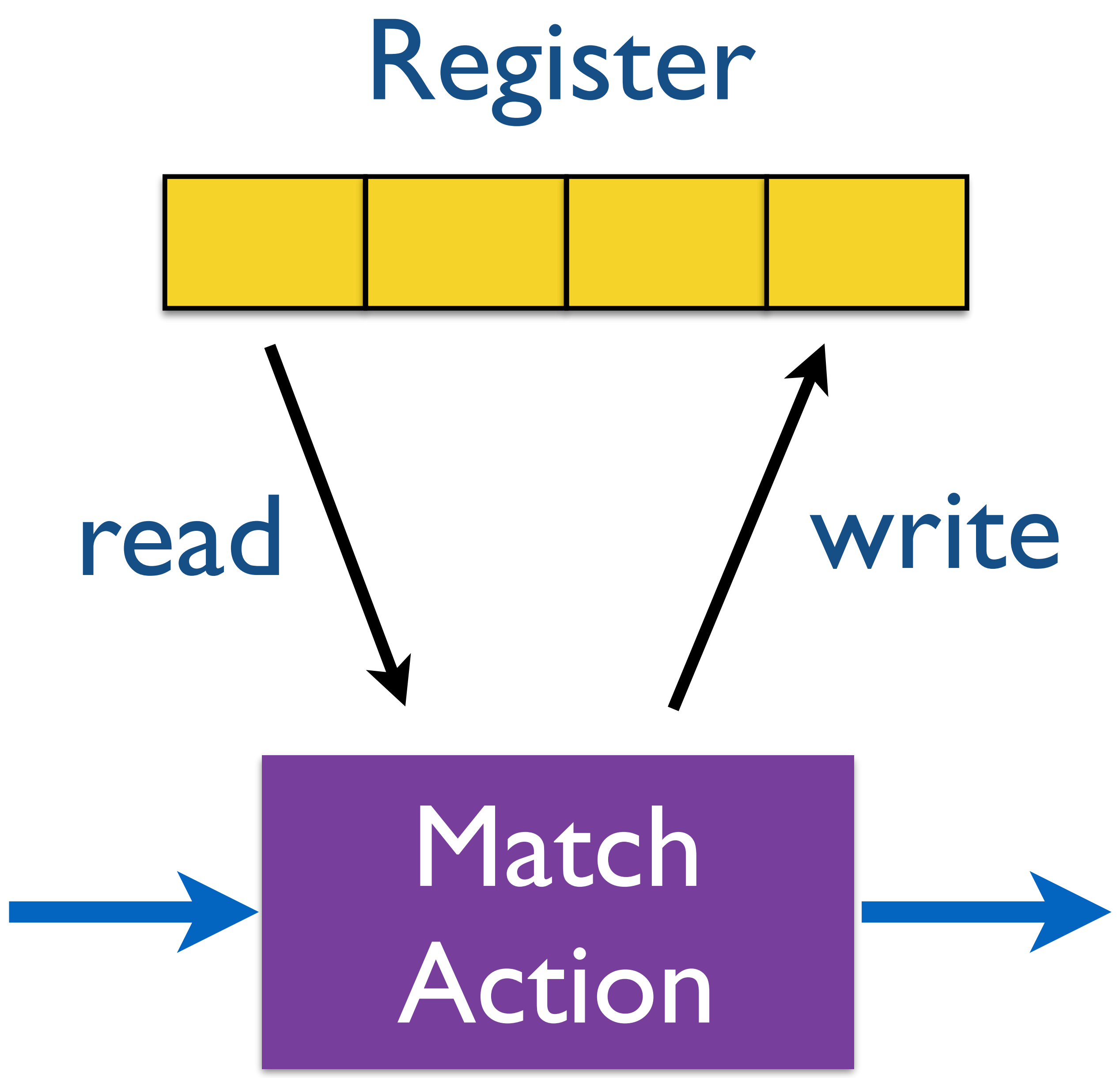
Packets always match and pass through all the tables

Parameters:

- Number of tables
- Checksum on / off



State Accesses



P4 doesn’t specify a concurrency model for state access

Performance depends on State Accesses Implementation

Parameters:

- Number of reads/writes to same register
- Number of reads/writes to different registers

Packet Modification

A single match-action table with a default action

The default action consists of an increasing number of add/remove header operations

Parameters:

- Number of add header operations
- Number of remove header operations

Platform-Specific Benchmark

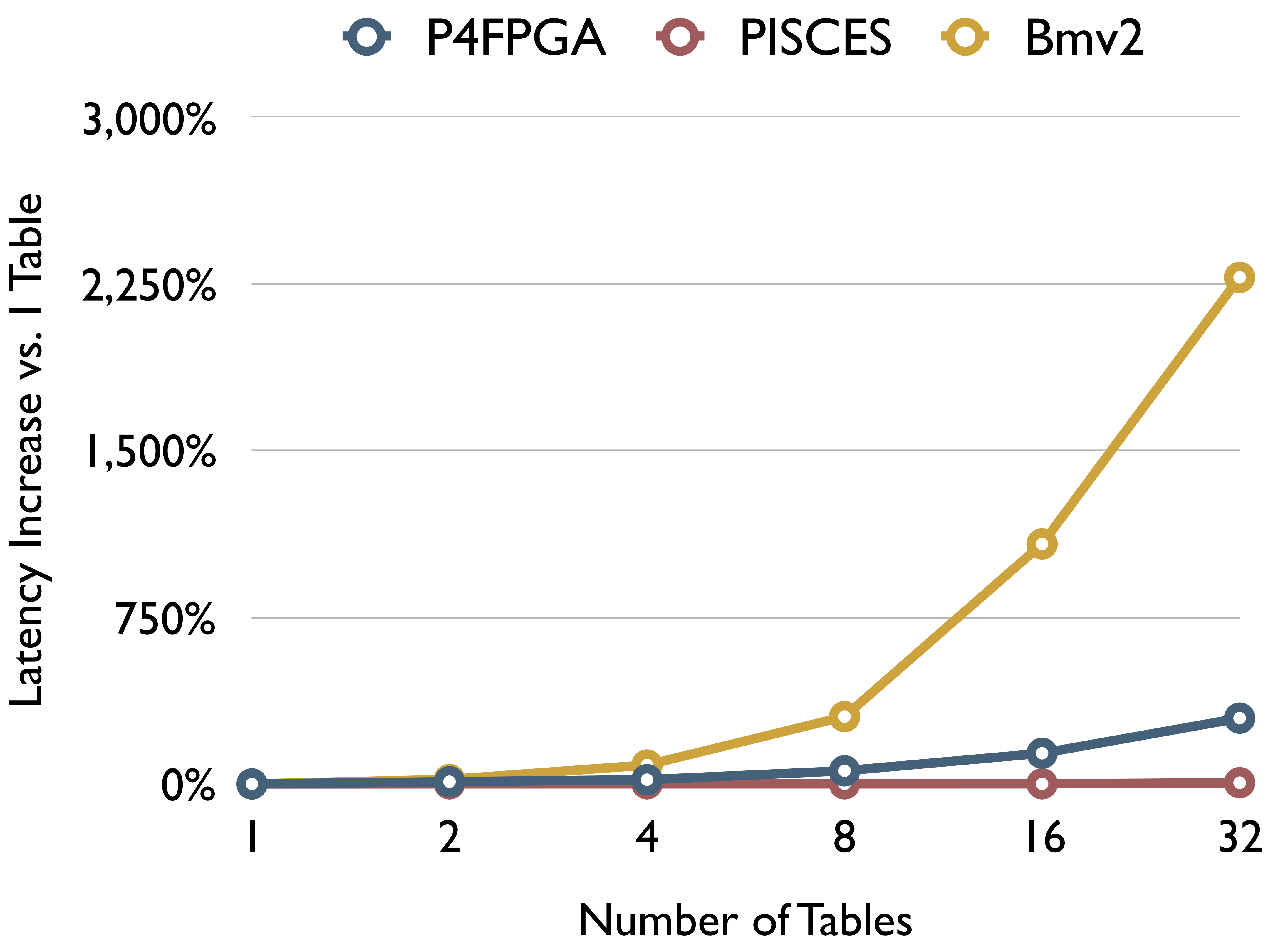
Target	Metric	Parameter
CPUs & NPUs	Latency Throughput	Changing Workflows Read/Write Same Register
FPGAs	Area Timing Resources	#Tables Size of tables
ASICs	Area Timing Resources Power	#Tables Size of tables #Depth of dependencies

Example Use Cases

Experimented with four P4 targets:

- P4c & Behavioral Model Switch (Bmv2)
- PISCES: customized OVS to support P4
- P4FPGA: compiled P4 for FPGAs (experimented with NetFPGA SUME board)
- Xilinx SDNet: compiled P4 for FPGAs (experimented with UltraScale+ XCVU13P)

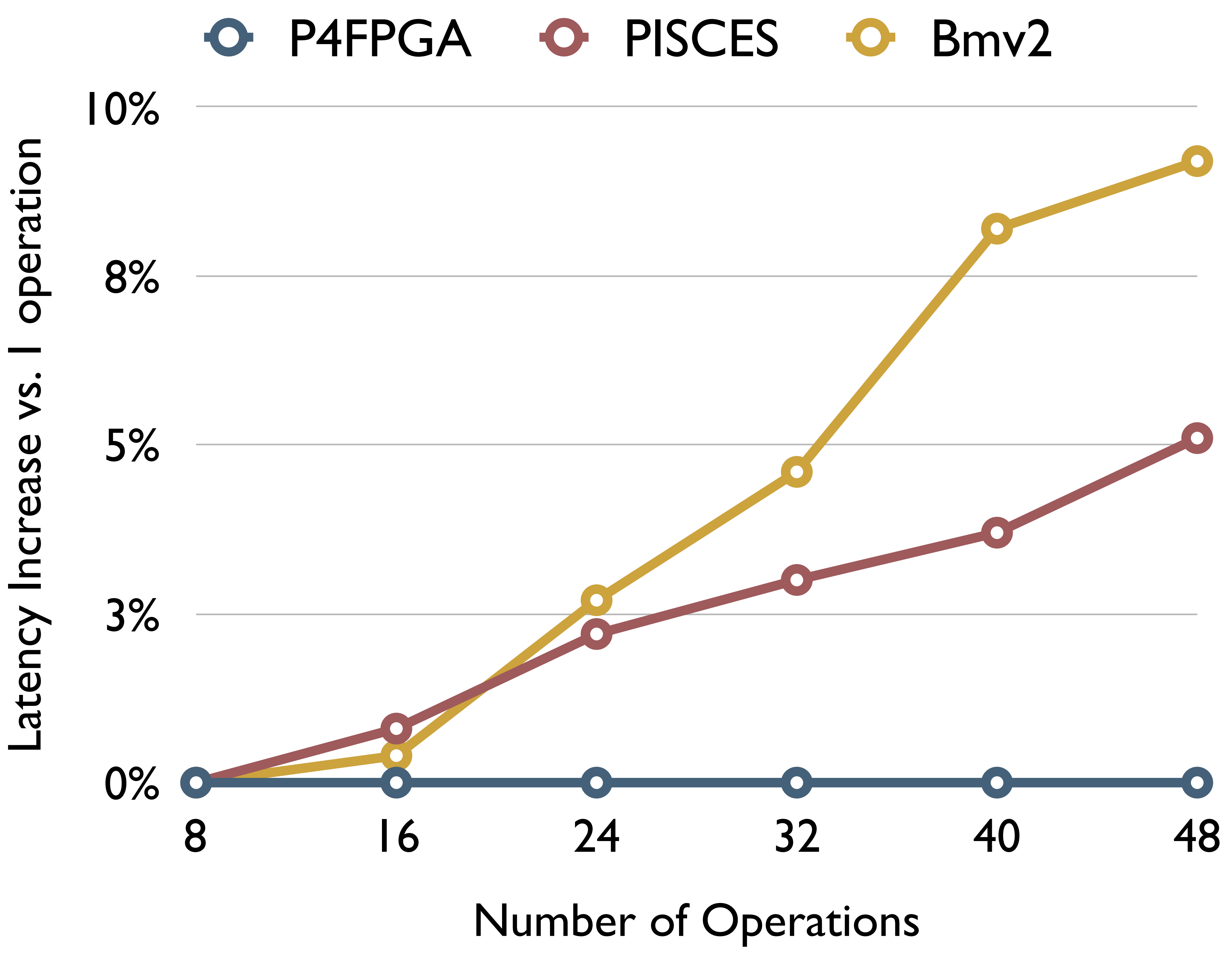
Benchmark Processing Pipeline



Results are normalized to the latency of applying a table

Tables in PISCES are converted to a big table

Benchmark Action Complexity

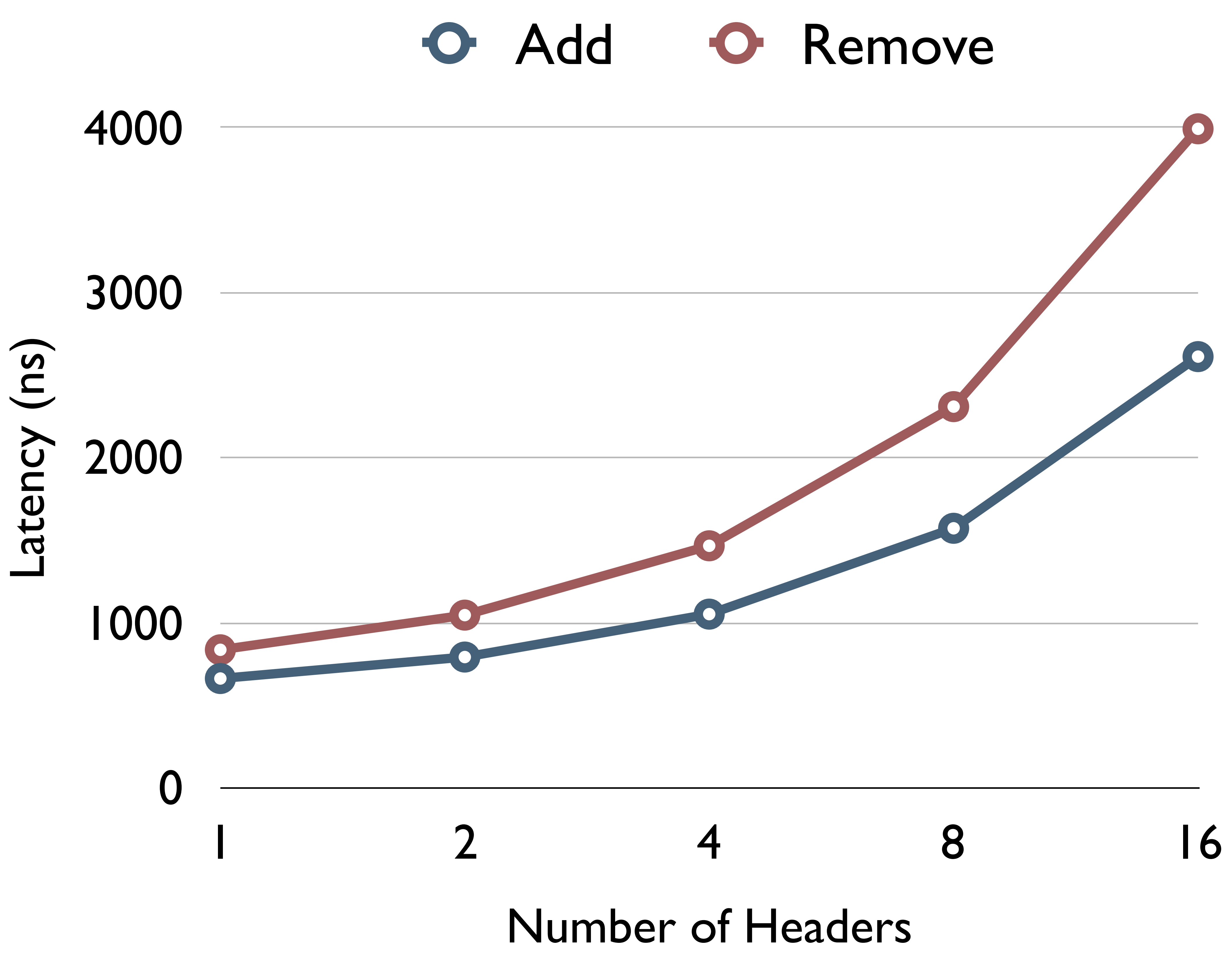


Results are normalized to the latency of an operation

P4FPGA schedules independent operations in a clock cycle

Bmv2 and PISCES execute field write operations sequentially

Benchmark Packet Modification



Experimented with $P4_{14}$-to-PX Xilinx SDNet on XCVU13P

Each header removal adds one stage

All header additions results in one stage

This behavior doesn't exist in $P4_{16}$-to-PX Xilinx SDNet

In Summary…

Whippersnapper: A synthetic P4 benchmark

- Addresses the need for a common criteria
- Evaluates key P4 language components
- Helps spur innovation

Try P4Benchmark

Install:

pip install p4benchmark

Generate P4 programs:

* p4benchmark --feature add-header --headers 2
* p4benchmark --feature set-field --operations 2

Questions?

For more details:

p4benchmark.org

Università
della
Svizzera
italiana

XILINX®

BAREFOOT NETWORKS

Huynh Tu Dang
huynh.tu.dang@usi.ch