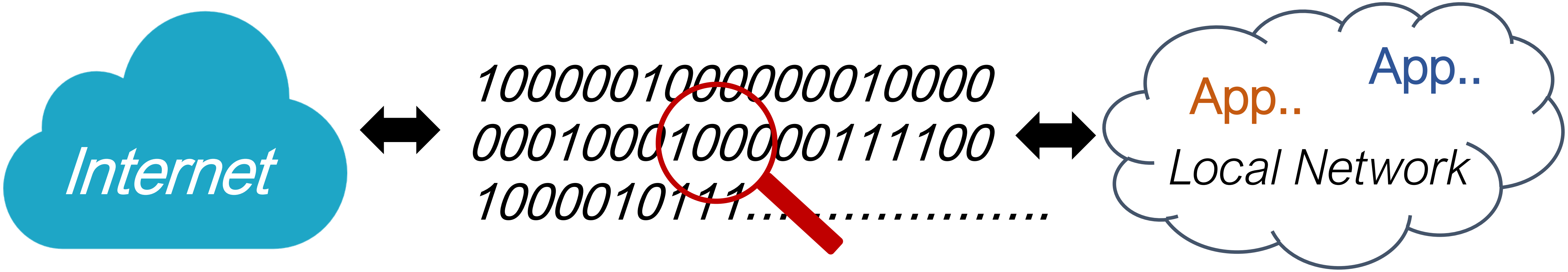
Achieving 100Gbps Intrusion Prevention on a Single Server

Zhipeng Zhao, Hugo Sadok, Nirav Atre, James C. Hoe, Vyas Sekar, Justine Sherry

Carnegie
Mellon
University

Intrusion Detection and Prevention System

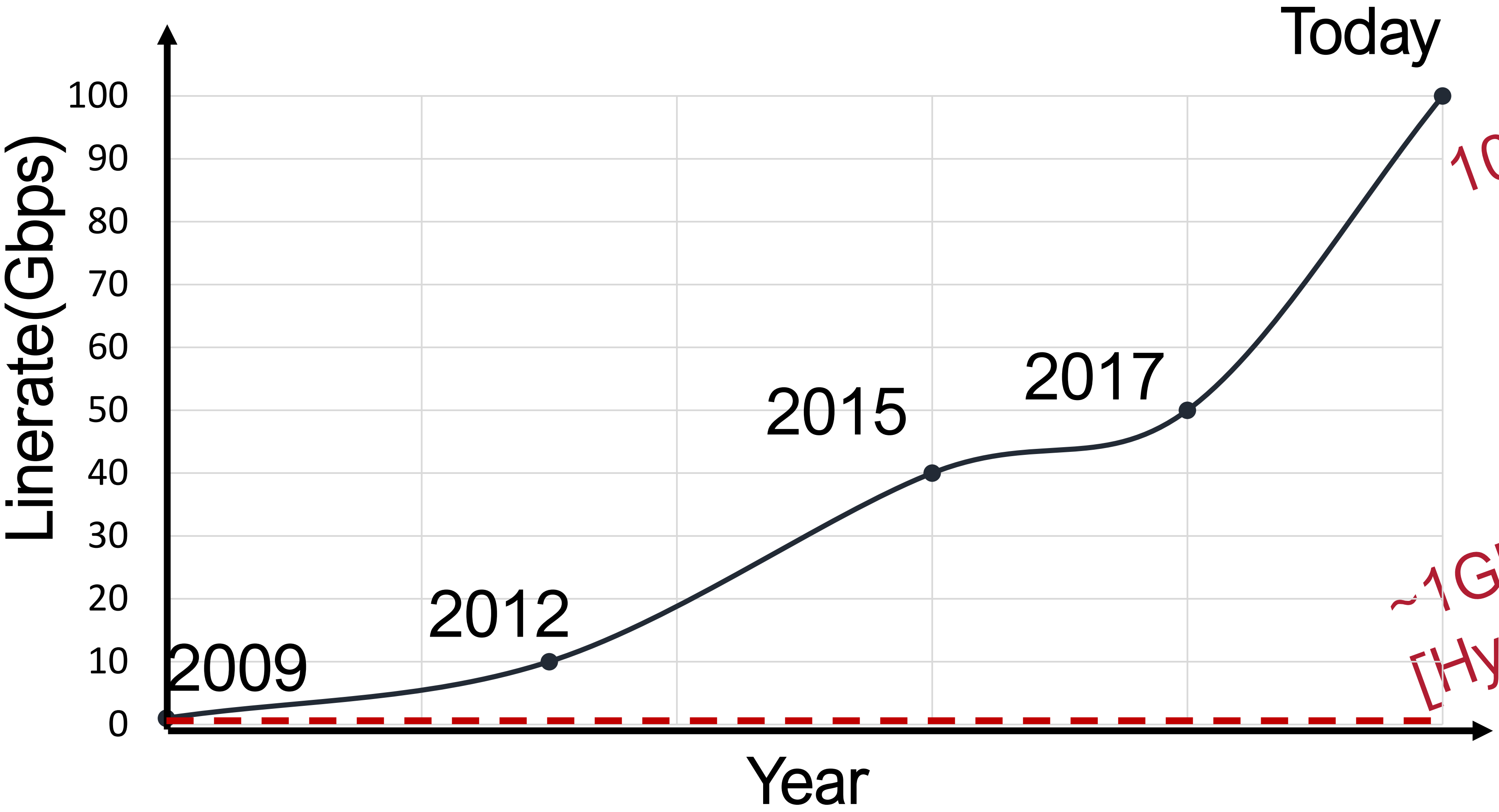


Intrusion Detection and Prevention System

- IDS/IPS is deployed at the gateway to identify network threats
- Check packets (including payload) against complex rules
- Compute intensive

Problem: State-of-the-Art Cannot Keep Up

Line Rate Evolution



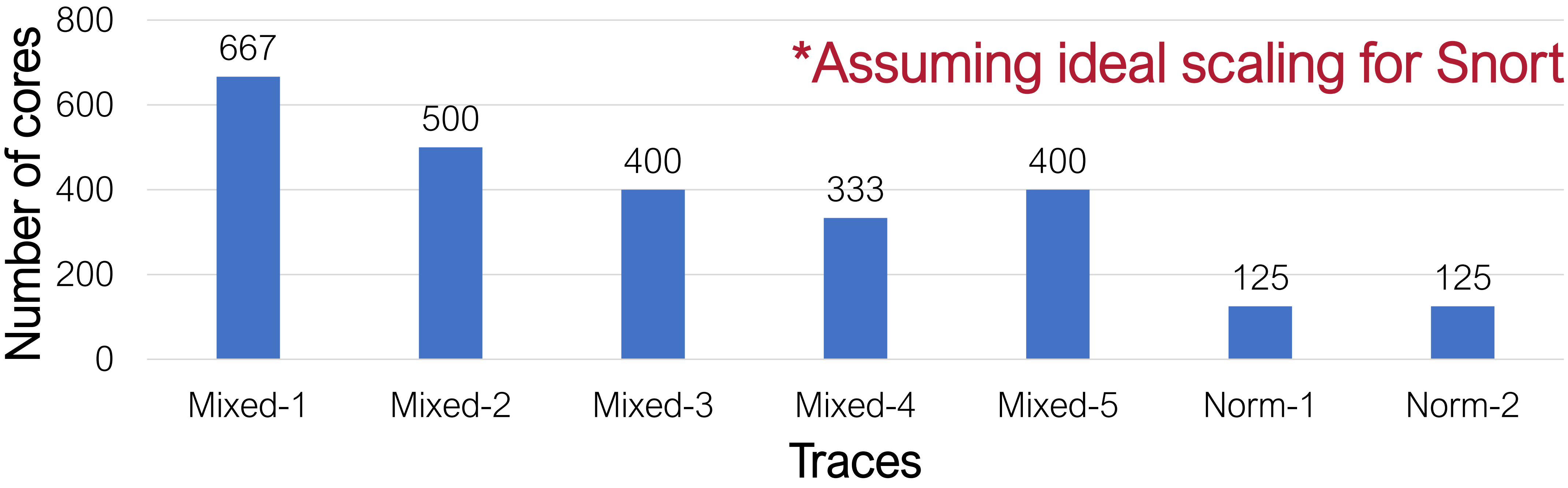
100 Gbps line rate!

~1Gbps per core
[Hyperscan, NSDI '19]

Inefficient to Scale Up Using State-of-the-Art

- Evaluate Snort 3.0 equipped with Hyperscan pattern matching library
- Need 4-21 servers (32-core) and 1125-6000 W

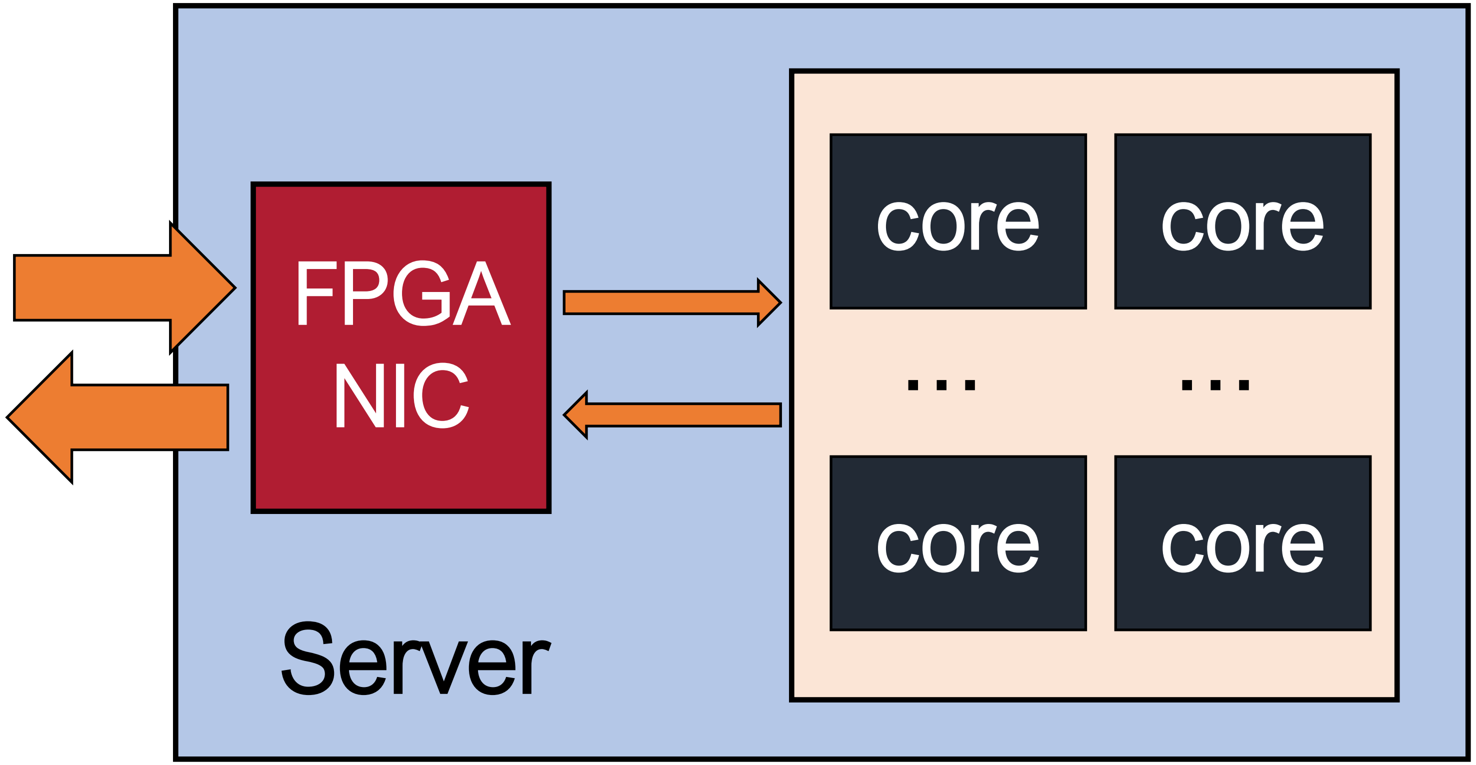
Number of Cores Needed to Reach 100Gbps



*Assuming ideal scaling for Snort

Pigasus: 100Gbps IPS on a Single Server

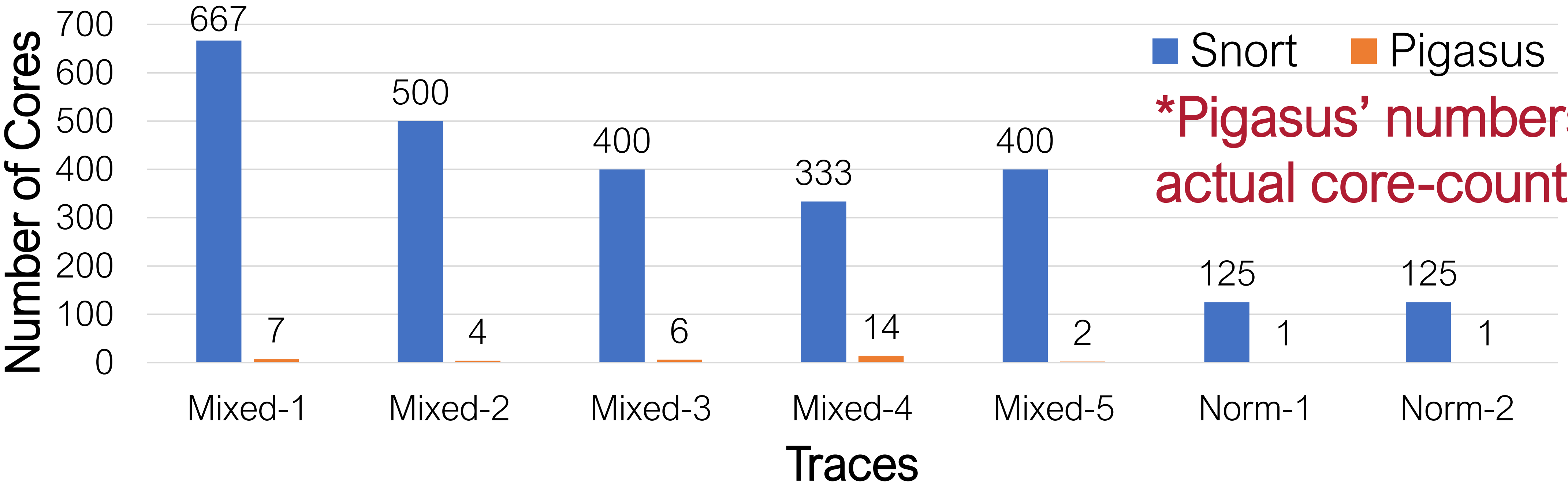
- 1 FPGA-based SmartNIC + 16-core CPU



Order-of-Magnitude Efficiency Improvement

- Snort: 4-21 servers (32-core) and 1125-6000 W
- Pigasus: 1 server (16-core) and 49-166 W

Number of Cores Needed to Reach 100Gbps



*Pigasus' numbers are actual core-count

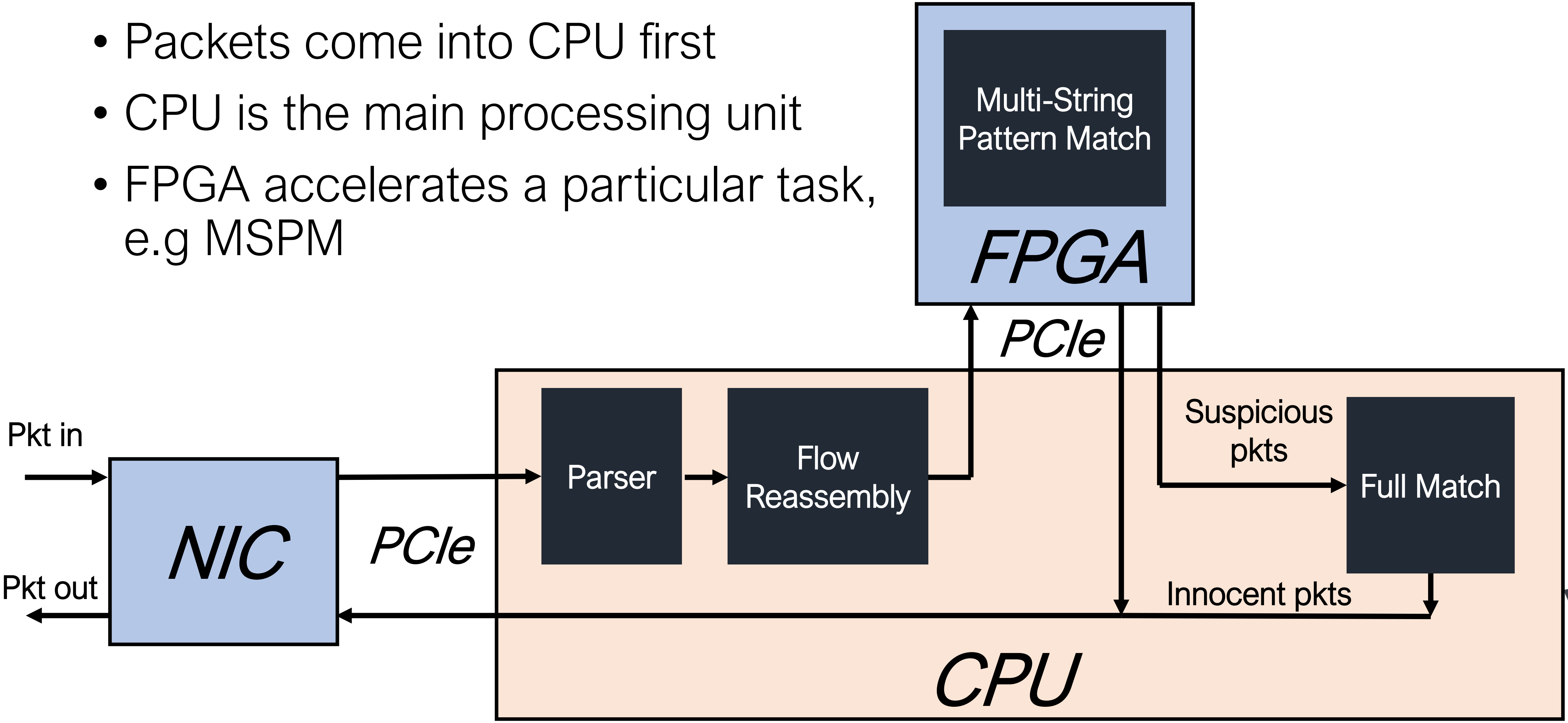
What is the secret sauce behind the 100x improvement?

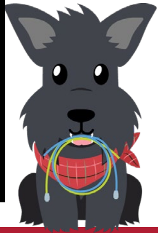
✓ FPGA-First Architecture

Fundamentally different scheme to make 100x improvement possible

Traditional “FPGA-as-Offload” Acceleration

- Packets come into CPU first
- CPU is the main processing unit
- FPGA accelerates a particular task, e.g MSPM

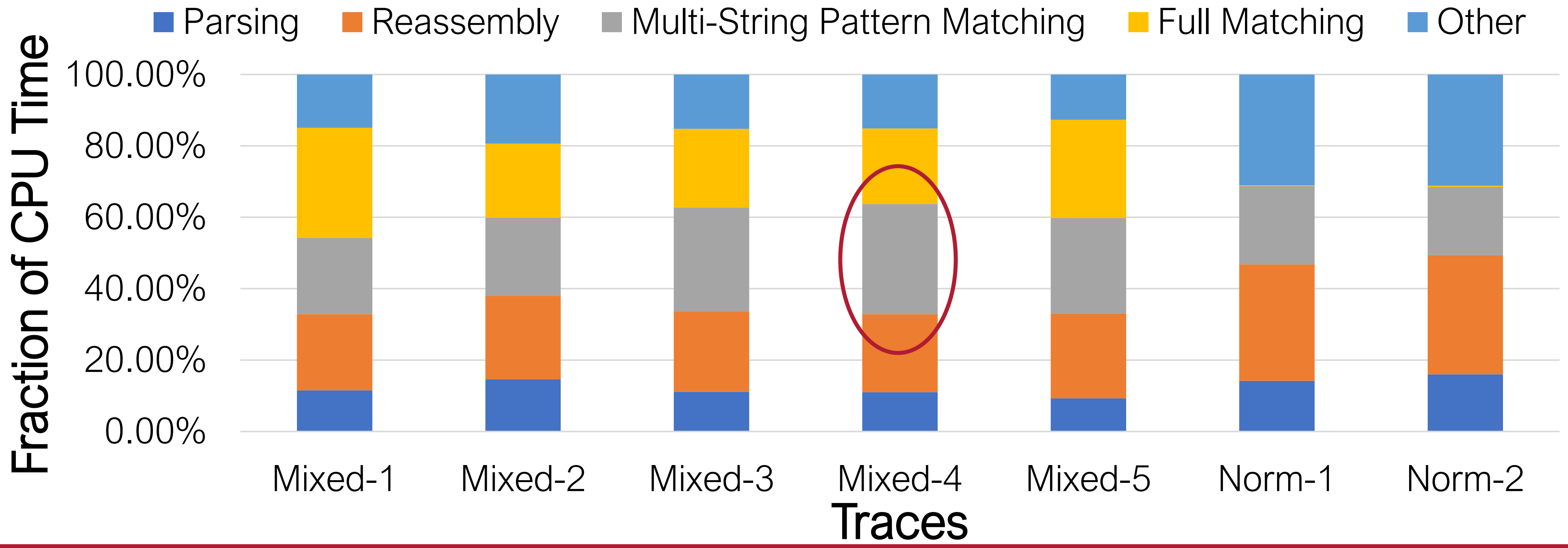




Prior Work Cannot Get 100x Speedup

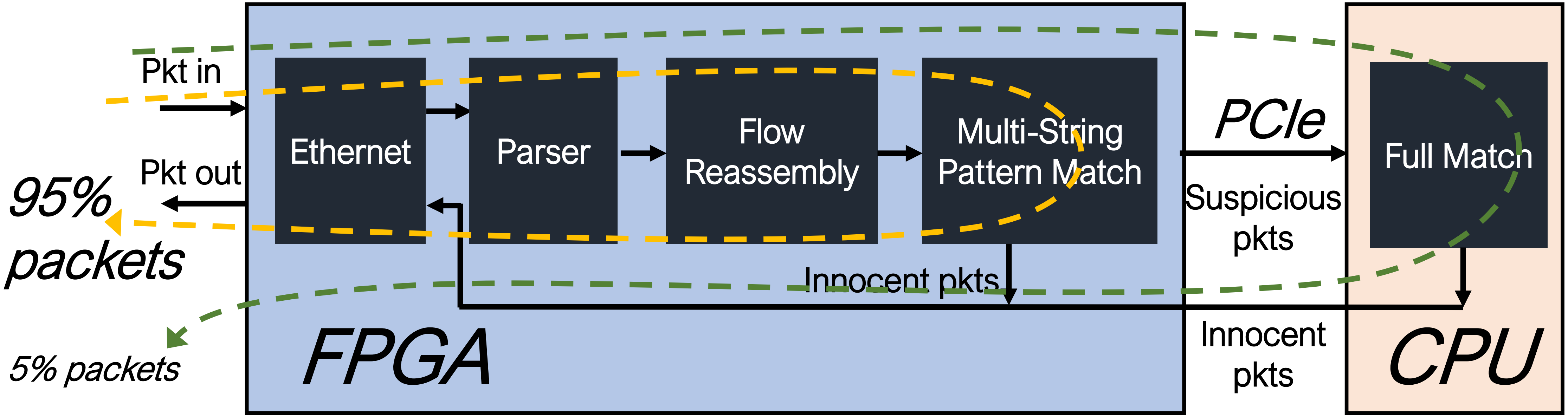
- No dominating task anymore (Hyperscan has made MSPM 8x faster)
- Up to ~2X speedup assuming ideal acceleration

Performance breakdown of Snort with Hyperscan



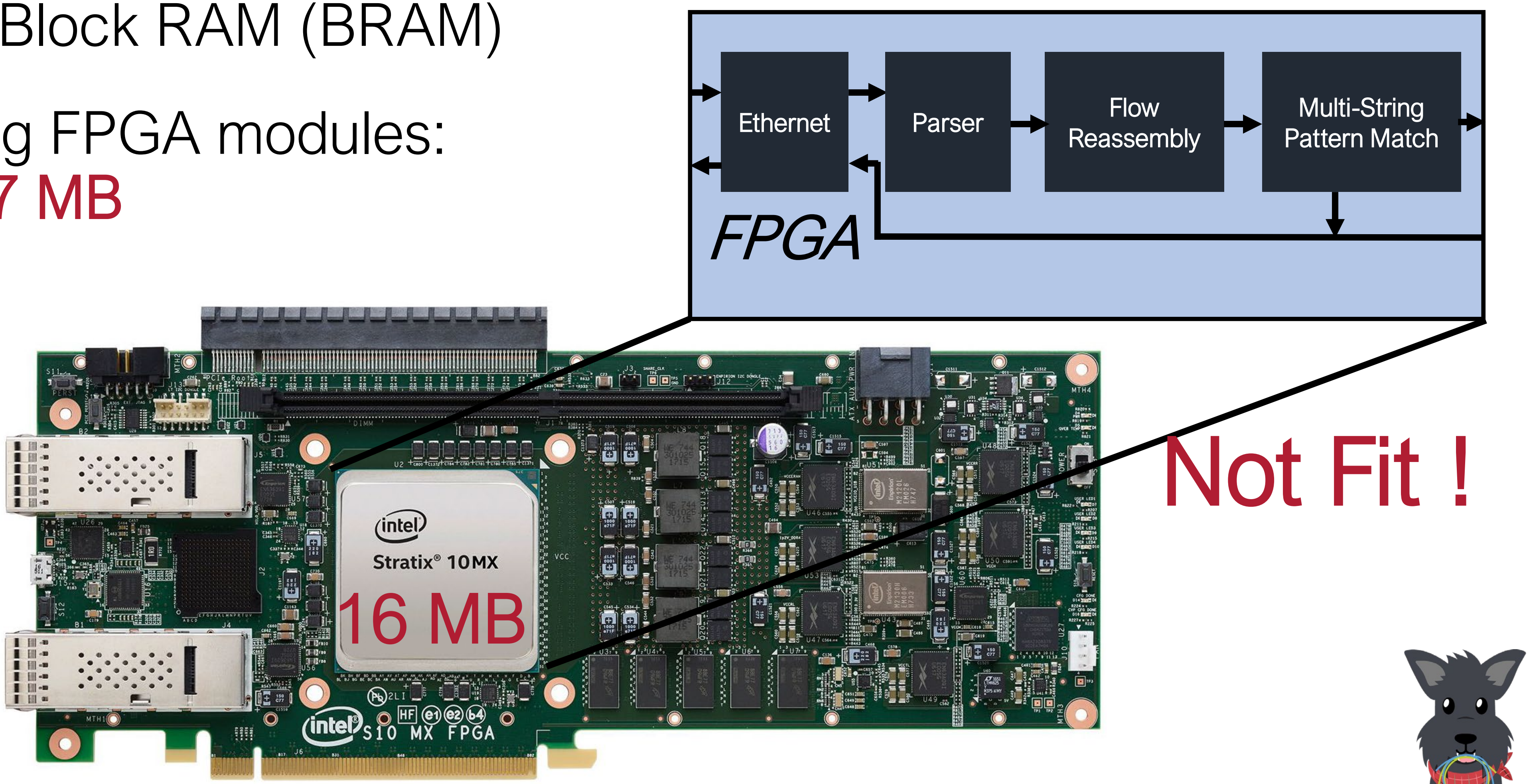
Pigasus: Inverted Offload Approach

- “FPGA-first” architecture: FPGA is the main processing unit
- Common cases are entirely processed on FPGA



Challenge: Limited Fast Memory on FPGA

- Only 16 MB Block RAM (BRAM)
- Using existing FPGA modules: more than 87 MB



Not Fit !

What is the secret sauce behind the 100x improvement?

✓ FPGA-First Architecture
Fundamentally different scheme to make 100x improvement possible

✓ Hierarchical Multi-String Pattern Matching (MSPM)
One of the algorithms to address the memory challenge

*Please refer to our paper for Flow Reassembly and Memory Resource Management

Multi-String Pattern Matching (MSPM)

- Checking payload and port number against 10K rules in “one” pass

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(…;content:”username=”,fast_pattern;
content:”/GetPermisssions.asp”;
pcre:”(^|&)username=[^&]*?”; sid = 2019;…)
```

Snort’s MSPM:

- Header
- Fast pattern

(Rest is checked by Full Matcher)

Pigasus’ MSPM:

- Header
- Fast pattern
- Non-fast string pattern

(Dominated Snort’s Full Matcher)

Any field mismatch => Rule not match

MSPM Design Options

State Machine-Based:
23 MB of BRAM

☹

Snort Hyperscan (Hashtable-based):
25 MB of BRAM

☹

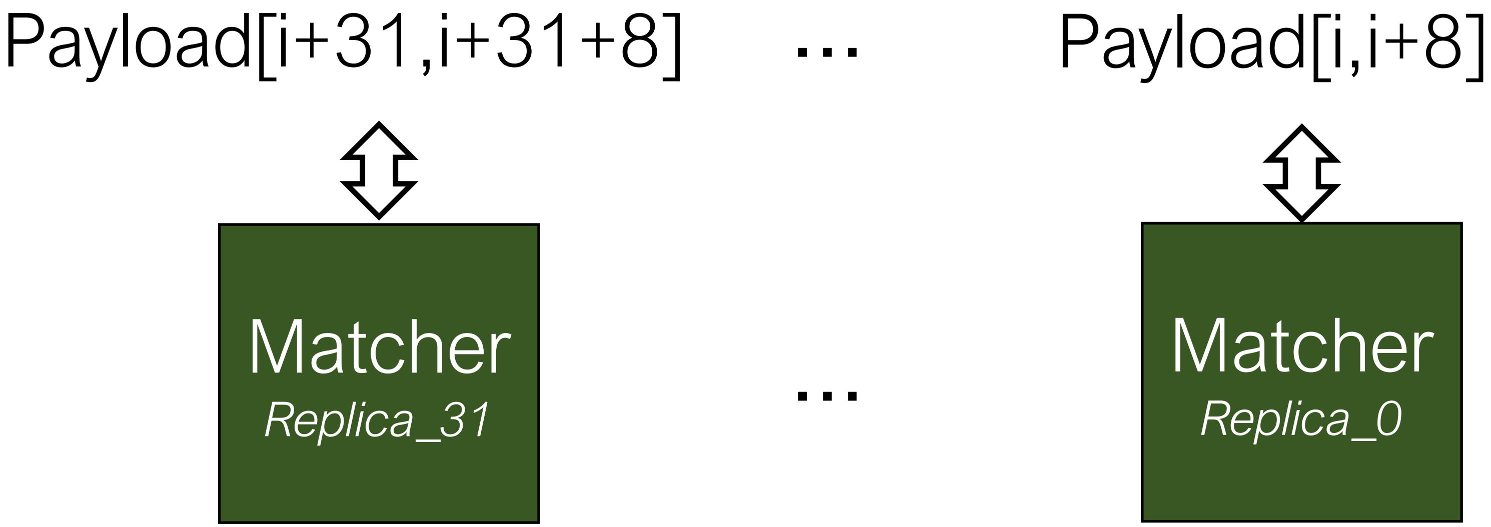
Complete more work

Pigasus (Hashtable-based):
3 MB of BRAM

☺

Pigasus Utilizes Perf & Memory Tradeoff

- High performance => Process more data in parallel => More memory



Payload[i+31,i+31+8] … Payload[i,i+8]

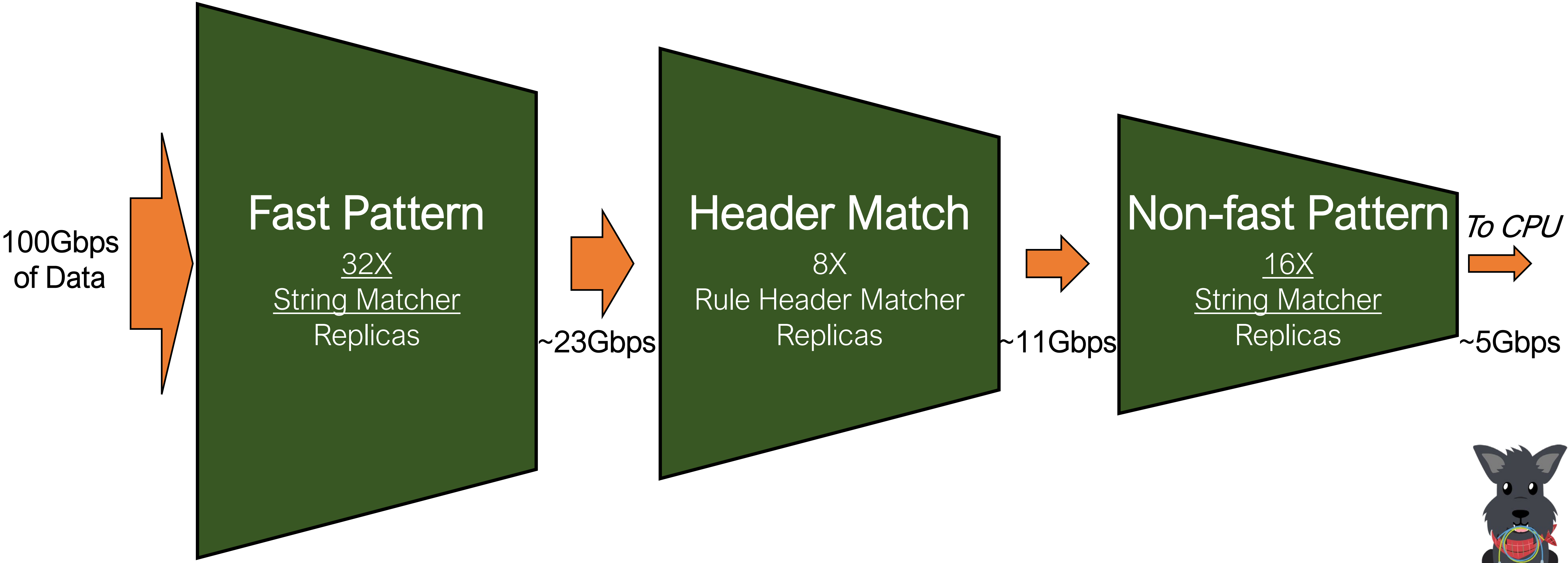
Matcher *Replica_31* … Matcher *Replica_0*

Key observation: no need to keep up 100Gbps ***everywhere***

Why not use less memory when lower performance is allowed

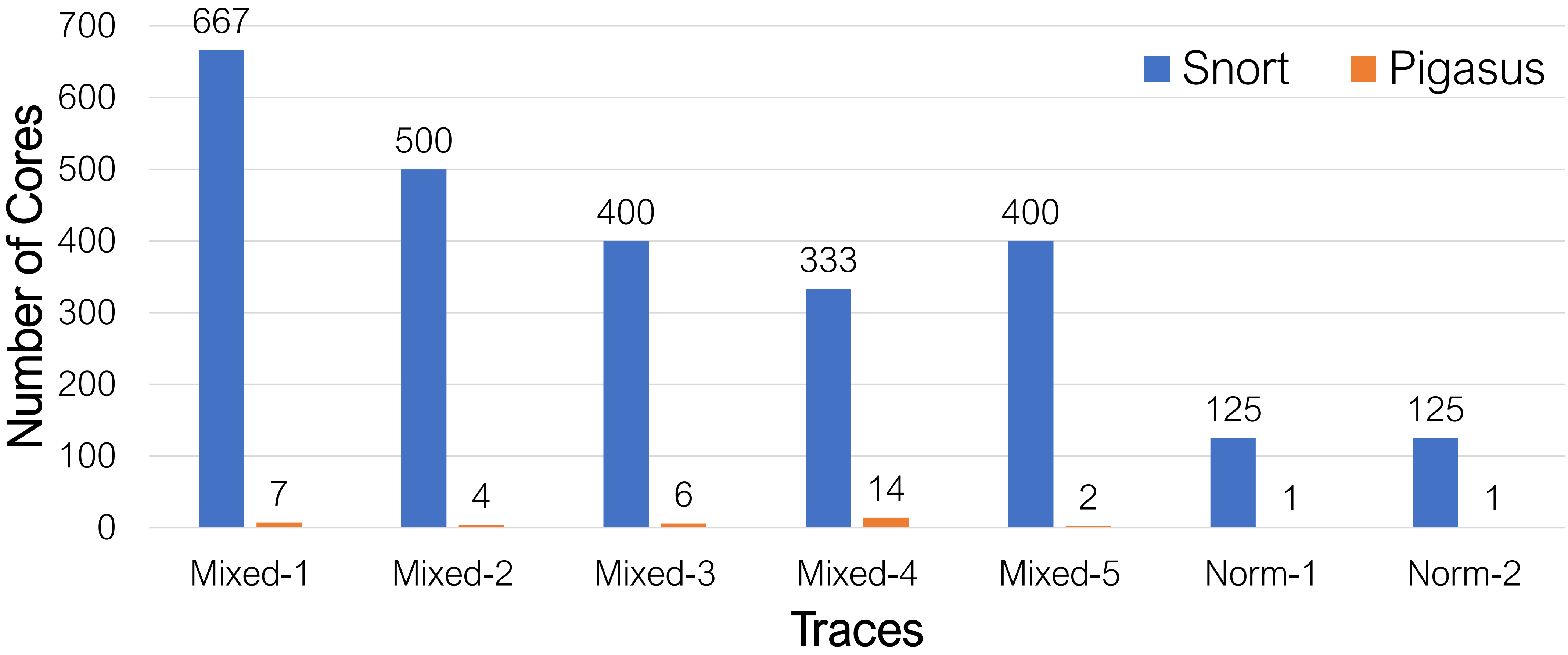
Use Hierarchical Filters to Save Memory

Key Idea: Hierarchical Filtering with Reduced Replicas at Each Layer



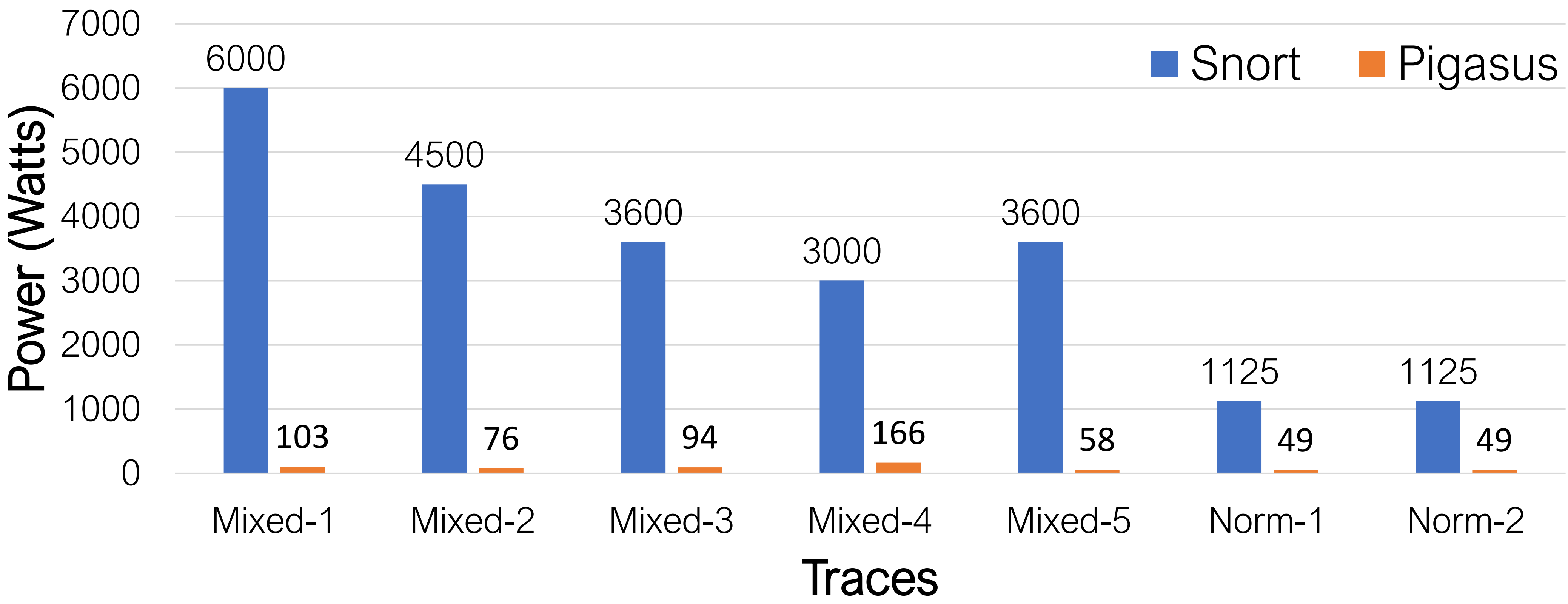
Evaluation

Pigasus Needs 100x Less Cores



*Snort's numbers are extrapolated from single core zero loss throughput
*Pigasus' numbers are actual core-count

Pigasus is Much Cheaper



Snort’s Total Cost of Ownership (TCO) $36,539

Pigasus’ TCO $10,642

*Assume 3 years lifetime

Conclusion

- Pigasus supports 100Gbps on a single server, saving hundreds of cores

- Pigasus proposes "FPGA-first" architecture, which is promising in performance but challenging to realize due to memory constraints

- Pigasus efficiently uses memory, e.g. Hierarchical Filtering in MSPM

Pigasus is publicly available at https://github.com/cmu-snap/pigasus

Contact: zhipengzhao@cmu.edu