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

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

- Eth
- IPv4
- IPv6
- UDP
- TCP
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

<table>
<thead>
<tr>
<th>Target</th>
<th>Metric</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs &amp; NPUs</td>
<td>Latency</td>
<td>Changing Workflows</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
<td>Read/Write Same Register</td>
</tr>
<tr>
<td>FPGAs</td>
<td>Area</td>
<td>#Tables</td>
</tr>
<tr>
<td></td>
<td>Timing</td>
<td>Size of tables</td>
</tr>
<tr>
<td></td>
<td>Resources</td>
<td></td>
</tr>
<tr>
<td>ASICs</td>
<td>Area</td>
<td>#Tables</td>
</tr>
<tr>
<td></td>
<td>Timing</td>
<td>Size of tables</td>
</tr>
<tr>
<td></td>
<td>Resources Power</td>
<td>#Depth of dependencies</td>
</tr>
</tbody>
</table>
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\textsubscript{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\textsubscript{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