P4All: Modular Switch Programming Under Resource Constraints

Mary Hogan*, Shir Landau-Feibish^, Mina Tahmasbi Arashloo+, Jennifer Rexford*, David Walker*

*Princeton University, ^The Open University of Israel, +Cornell University
Traditional switches hinder innovation
Protocol Independent Switch Architecture
Protocol Independent Switch Architecture

Intel® Tofino™

PENSANDO

Programmable Configuration

BROADCOM®

PISA switch
Programming Protocol
Independent Packet Processors

P4 Program

Programmable Configuration

PISA switch
Programming Protocol
Independent Packet Processors

P4 Program
- Measure heavy hitters
- Rate limiting
- Identify and mitigate attacks

Programmable Configuration

PISA switch
P4 code should be reusable
P4 code should be reusable
P4 code should be reusable
P4 code is not reusable
P4 code is not reusable

Data structures (e.g., hash tables, count-min sketch) are valid for a range of sizes
P4 code is not reusable

Data structures (e.g., hash tables, count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size (e.g., amount of memory used)
P4 code is not reusable

Data structures (e.g., hash tables, count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size (e.g., amount of memory used)

Switches have very limited resources that are shared across all program elements
P4 code is not reusable

Data structures (e.g., hash tables, count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size (e.g., amount of memory used)

Switches have very limited resources that are shared across all program elements

Commonly used data structures are rewritten often
P4 code is not reusable

Data structures (e.g., hash tables, count-min)

P4 makes it possible to program the network, but it does not make it easy.

Commonly used data structures are rewritten often
Circular Development

P4 Program
Circular Development

P4 Program

P4 Compiler
Circular Development

P4 Compiler

Program doesn't fit

P4 Program
Circular Development

P4 Program

Program doesn't fit

P4 Compiler

Program fits

Target
P4All mitigates circularity
P4All mitigates circularity

P4All streamlines development by allowing for reusable **elastic** data structures
P4All mitigates circularity

P4All streamlines development by allowing for reusable **elastic** data structures

Elastic data structures are defined by symbolic values that stretch or shrink as needed
P4All mitigates circularity

P4All streamlines development by allowing for reusable elastic data structures

Elastic data structures are defined by symbolic values that stretch or shrink as needed

P4All automatically sizes programs to make optimal use of available switch resources
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
Protocol-Independent Switch Architecture
PISA
PISA

Programmable Parser

Packet Header Vector

Pipeline Stages
PISA

Programmable Parser

Packet Header Vector

ALU

Pipeline Stages
PISA

Programmable Parser

Packet Header Vector

Persistent State

ALU

Pipeline Stages
PISA

Programmable Parser

Packet Header Vector

Persistent State

ALU

Pipeline Stages

Programmable Deparser
Packet Header Vector

Persistent State

ALU

Pipeline Stages
Data structure
PISA
The shapes of data structures change based on the application.
Count-Min Sketch

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
Count-Min Sketch

```
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```
Count-Min Sketch

\[
x \xrightarrow{h_1(x)} 1 \quad 0 \quad 0 \quad 0
\]
\[
x \xrightarrow{h_2(x)} 0 \quad 1 \quad 0 \quad 0
\]
\[
x \xrightarrow{h_3(x)} 1 \quad 0 \quad 0 \quad 0
\]
Count-Min Sketch

<table>
<thead>
<tr>
<th>h_1(y)</th>
<th>h_2(y)</th>
<th>h_3(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Count-Min Sketch

Count(x) = 1
Data Plane Caching
Data Plane Caching

Cache of popular keys

Server cache
Data Plane Caching

Key 1

Cache of popular keys

Server cache
Data Plane Caching

Cache of popular keys

Key 1

Server cache
Data Plane Caching

Cache of popular keys

Server cache

Value
# Data Plane Caching

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
</tbody>
</table>

Cache of popular keys
## Tracking Key Popularity

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
</tbody>
</table>

The key popularity is represented in the cache and CMS. Each key's popularity is indicated by the value associated with it in the cache.
## Tracking Key Popularity

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- **Cache of popular keys**
- **CMS**

If requests for key > 80, insert into cache
PISA

Cache → Cache → CMS → CMS
PISA

How to size the data structures?
Resources vs Accuracy

Actual count(x) = 50

Estimated count(x) = 80
Resources vs Accuracy

Actual count(x) = 50

Estimated count(x) = 60
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
Symbolic Values

rows = 3

cols = 4
Symbolic Values

\[
\begin{align*}
\text{cols} &= 4 \\
\text{rows} &= 3 \\
\text{register} &<\text{bit}^{32}>^{(4)} \text{ row1;} \\
\text{register} &<\text{bit}^{32}>^{(4)} \text{ row2;} \\
\text{register} &<\text{bit}^{32}>^{(4)} \text{ row3;}
\end{align*}
\]
Symbolic Values

(rows = ?)  

(cols = ?)  

[Diagram showing a grid with placeholders for rows and cols]
Symbolic Values

symbolic rows;
symbolic cols;
register<bit<32>>(cols)[rows] cms_rows;

rows = ?
cols = ?
For Loops
For Loops

increment_row1();
increment_row2();
increment_row3();

\[ h_1(x) \]
\[ h_2(x) \]
\[ h_3(x) \]
For Loops

\[ h_1(x) \]

\[ h_{\text{rows}}(x) \]

\[
\begin{array}{c}
\text{x} \\
1 \\
\vdots \\
1 \\
\vdots \\
\ddots \\
\vdots \\
1 \\
\vdots \\
\end{array}
\]
For Loops

```java
for (i < rows) {
    increment_row()[i];
}
```

\[
h_1(x) \rightarrow \begin{array}{c}
1 \\
\hline
x \\
\hline
\end{array} \rightarrow \begin{array}{c}
h_{rows}(x) \\
\hline
1 \\
\hline
\end{array} \rightarrow \begin{array}{c}
\ldots \\
\hline
\end{array}
\]
Objective Functions

\[ f(\text{cols}) = \text{CMS error} \]
Objective Functions

\texttt{objective cms\_error \{ f(cols) \} }
\texttt{minimize cms\_error;}

\( f(\text{cols}) = \text{CMS error} \)
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
P4All Compiler

P4All Program + Target Specification (resource constraints, etc.)

Concrete values for symbolic values (P4 Program) + Mapping from program elements to pipeline stages
P4All Compiler

Target Specification
(resource constraints, etc.)

P4All Program

+ 

P4All Compiler

Generate and Solve Integer-Linear Program (ILP)

Concrete values for symbolic values (P4 Program) + Mapping from program elements to pipeline stages
ILP Constraints

Packet Header Vector

Alu

Persistent State

Pipeline Stages
ILP Objective

\[
\text{objective } \text{cms\_error} \{ f(\text{cols}) \} \\
\text{minimize } \text{cms\_error};
\]

\[
f(\text{cols}) = \text{CMS error}
\]
P4All Compiler

symbolic rows = 6
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
# P4All Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Compile Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td>1.8</td>
</tr>
<tr>
<td>Key-value store</td>
<td>15.4</td>
</tr>
<tr>
<td>Key-value store + CMS</td>
<td>27.9</td>
</tr>
<tr>
<td>Switch.p4</td>
<td>0.2</td>
</tr>
<tr>
<td>IP forwarding + stateful firewall</td>
<td>0.4</td>
</tr>
<tr>
<td>Beaucoup</td>
<td>0.1</td>
</tr>
<tr>
<td>Precision</td>
<td>25.7</td>
</tr>
<tr>
<td>NetChain</td>
<td>27.9</td>
</tr>
<tr>
<td>SketchLearn</td>
<td>2.4</td>
</tr>
<tr>
<td>Conquest</td>
<td>5.8</td>
</tr>
</tbody>
</table>
ILP Performance
ILP Performance

![Bar chart showing the performance of different ILP components at various stages of Tofino. The chart compares the number of ILP components across different stages, divided into categories: Dependency Constraints, Other Constraints, Resource Constraints, and Variables.](chart.png)
ILP Performance

![Bar chart showing the performance of Tofino stages with different multipliers (1.8s, 4.5s, 53s, 216s). The chart compares the time taken for different ILP components: Dependency Constraints, Other Constraints, Resource Constraints, and Variables. The x-axis represents the stages of Tofino, and the y-axis shows the ILP components.](image)
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Ongoing + Future Work
Ongoing + Future Work

Design representative objective functions
Ongoing + Future Work

Design representative objective functions

Object-oriented programming model
Ongoing + Future Work

Design representative objective functions

Object-oriented programming model

Query language abstraction
P4All: Modular Switch Programming Under Resource Constraints

Mary Hogan, Shir Landau-Feibish, Mina Tahmasbi Arashloo, Jennifer Rexford, David Walker

mh43@cs.princeton.edu