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

Fixed-function switch
Protocol Independent Switch Architecture

Programmable Configuration

PISA switch
Protocol Independent Switch Architecture
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 Program
- P4 Compiler
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
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

Conclusion
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Conclusion
Protocol-Independent Switch Architecture
PISA
PISA
PISA

Programmable Parser

Packet Header Vector

ALU

Pipeline Stages
PISA
PISA

Packet Header Vector

Persistent State

ALU

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

The Count-Min Sketch is a probabilistic data structure that can be used to estimate the frequency of elements in a data stream. It is particularly useful for approximate counting and can be implemented efficiently on machines with limited memory.

## Example

Consider a simple example where we have a data stream containing the following elements: 1, 0, 0, 0, 1, 0, 0, 0. We can use three hash functions, $h_1(x)$, $h_2(x)$, and $h_3(x)$, to map each element to a set of counters. For instance, if we use $h_1(x) = x$, $h_2(x) = x^2$, and $h_3(x) = x^3$, the resulting table would look as follows:

<table>
<thead>
<tr>
<th>Element</th>
<th>$h_1(x)$</th>
<th>$h_2(x)$</th>
<th>$h_3(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The Count-Min Sketch algorithm maintains the minimum value across all the counters for each element, providing an upper bound on its frequency. This makes it an efficient tool for estimating the frequency of elements in a streaming data setting.
Count-Min Sketch

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Count-Min Sketch

Count(x) = 1
Data Plane Caching

NetCache, Jin et al. [SOSP'17]
Data Plane Caching

Cache of popular keys

Key-value store

NetCache, Jin et al. [SOSP’17]
Data Plane Caching

Cache of popular keys

Key-value store

NetCache, Jin et al. [SOSP’17]
Data Plane Caching

Cache of popular keys

Key-value store

NetCache, Jin et al. [SOSP’17]
Data Plane Caching

Cache of popular keys

Key-value store

Value

NetCache, Jin et al. [SOSP’17]
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>

Cache of popular keys

Key 5

CMS

Values:
- 100
- 150
- 120
### 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>

If requests for key > 80, insert into cache
PISA
How to size the data structures?
## Resources vs Accuracy

<table>
<thead>
<tr>
<th></th>
<th>Actual count(x) = 50</th>
<th>Estimated count(x) = 80</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Resources vs Accuracy

Actual count(x) = 50

Estimated count(x) = 60
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Conclusion
Elastic Structures

rows = 3

cols = 4
Elastic Structures

register<bit<32>>(4) row1;
register<bit<32>>(4) row2;
register<bit<32>>(4) row3;
Elastic Structures
symbolic rows;
symbolic cols;
register<bit<32>>(cols)[rows] cms_rows;

Elastic Structures

cols = ?
rows = ?
Elastic Operations

\[ x \xrightarrow{h_1(x)} 1 \xrightarrow{h_{\text{rows}}(x)} \ldots \]

\[ 1 \xrightarrow{\ldots} \]
for (i < rows) {
    increment_row()[i];
}
Objective Functions

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

\[
\begin{align*}
\text{objective } & \text{cms\_error } \{ f(\text{cols}) \} \\
\text{minimize } & \text{cms\_error;}
\end{align*}
\]

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

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Conclusion
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
symbolic rows = 6

P4All Compiler

CMS row 1
CMS row 2
CMS row 3
CMS row 4
CMS row 5
CMS row 6

CMS row 7
CMS row 8
P4All Compiler

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

P4All Compiler

Generate and Solve Integer-Linear Program (ILP)

Concrete values for symbolic values (P4 Program) + Mapping from program elements to pipeline stages
Packet Header Vector

Persistent State

ALU

ILP Constraints

Pipeline Stages
ILP Objective

```
objective cms_error { f(cols) }
minimize cms_error;
```

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

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Conclusion
## 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 Overhead
Outline

Elastic Structures

P4All
  Language
  Compiler
  Evaluation

Conclusion
Conclusion

Elastic data structures expand to use the available resources
Conclusion

Elastic data structures expand to use the available resources

The P4All compiler finds the optimal structure size for specific applications
Conclusion

Elastic data structures expand to use the available resources

The P4All compiler finds the optimal structure size for specific applications

Reusable modules in P4All make it easier to implement and deploy data-plane applications
P4All: Modular Switch Programming Under Resource Constraints

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

mh43@cs.princeton.edu