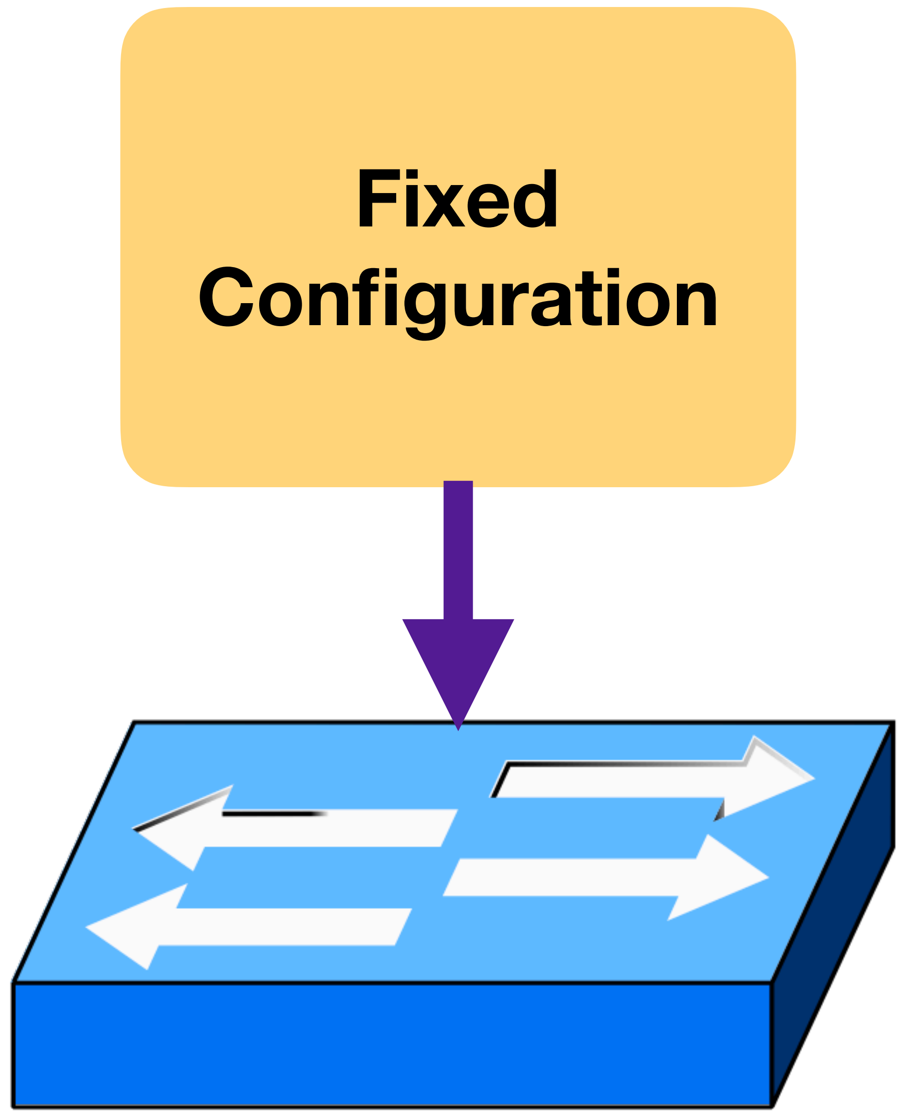
P4All: Modular Switch Programming Under Resource Constraints

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

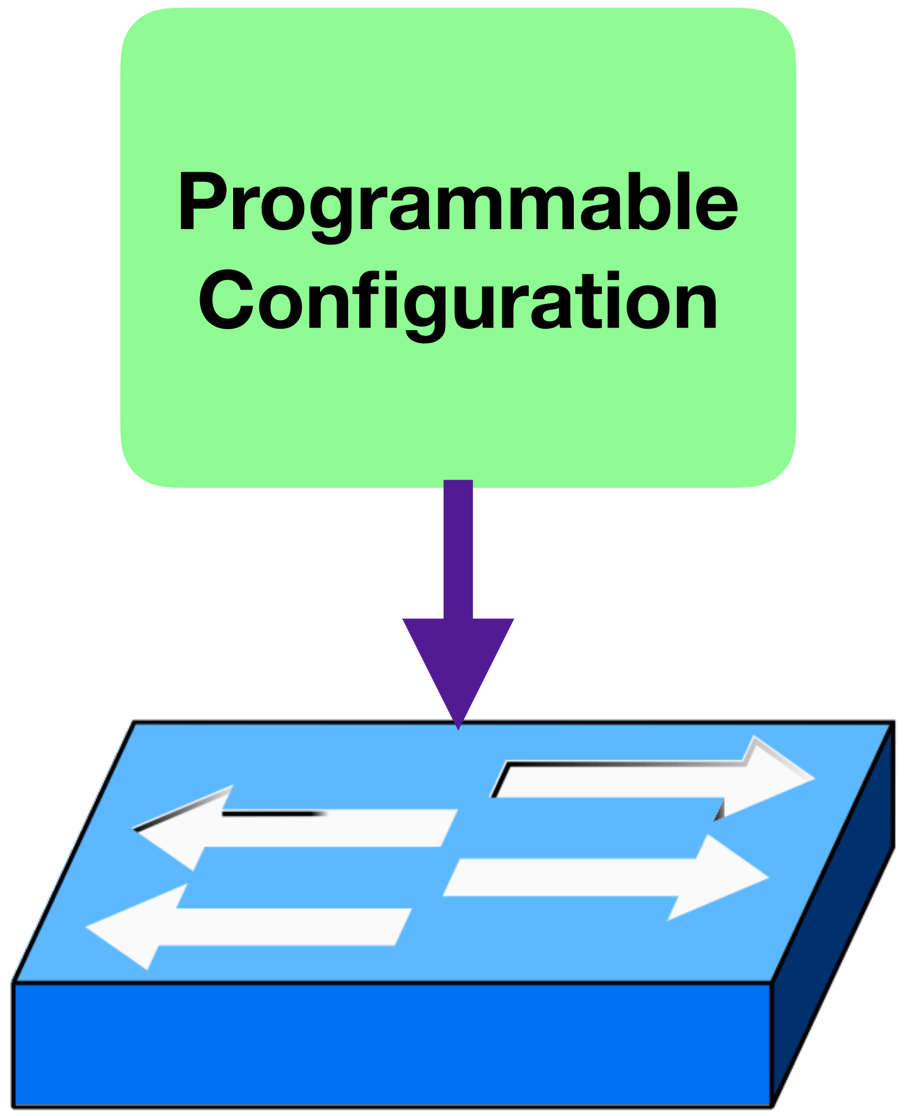
*Princeton University, ^The Open University of Israel, +Cornell University

Traditional switches hinder innovation



Fixed-function switch

Protocol Independent Switch Architecture



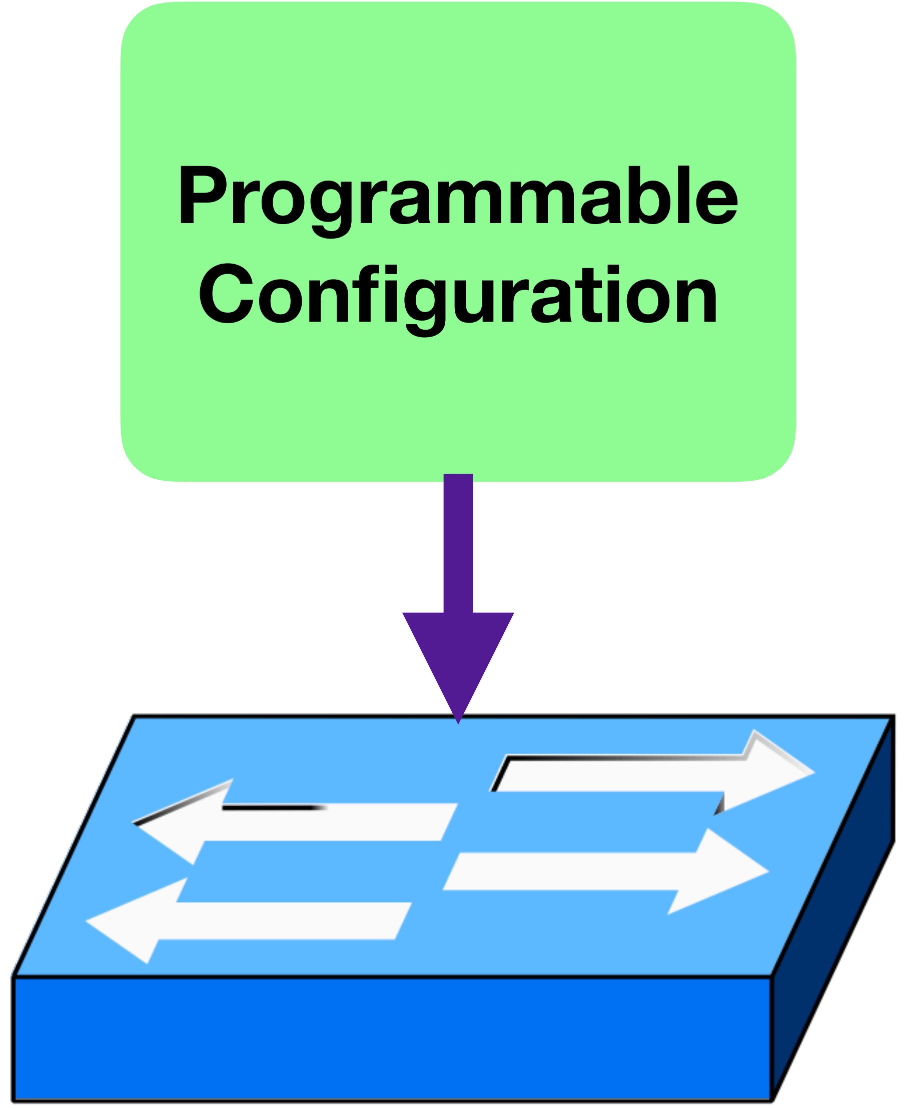
Programmable Configuration

PISA switch

Protocol Independent Switch Architecture

Intel® Tofino™

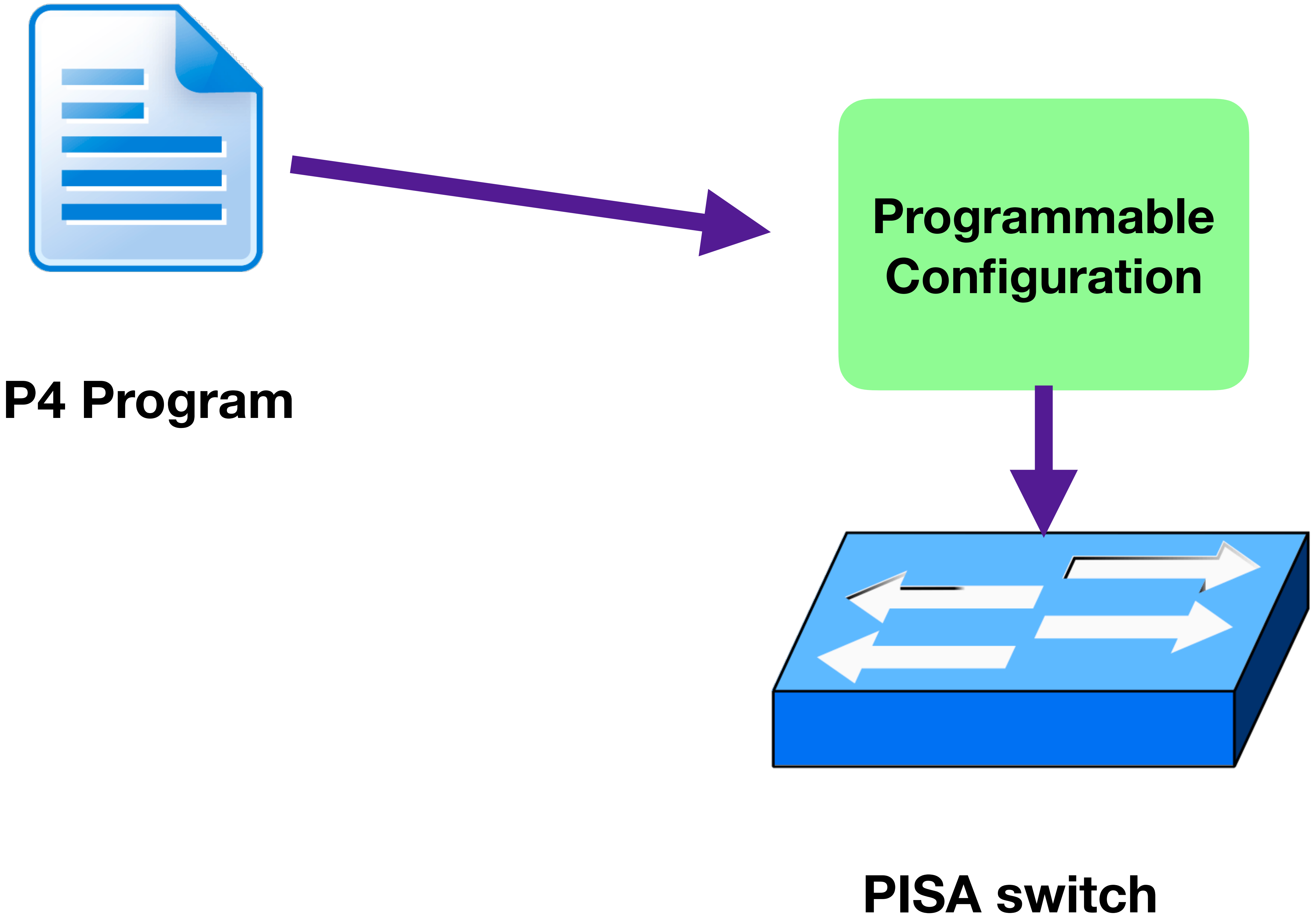
PENSANDO



PISA switch

BROADCOM®

Programming Protocol Independent Packet Processors

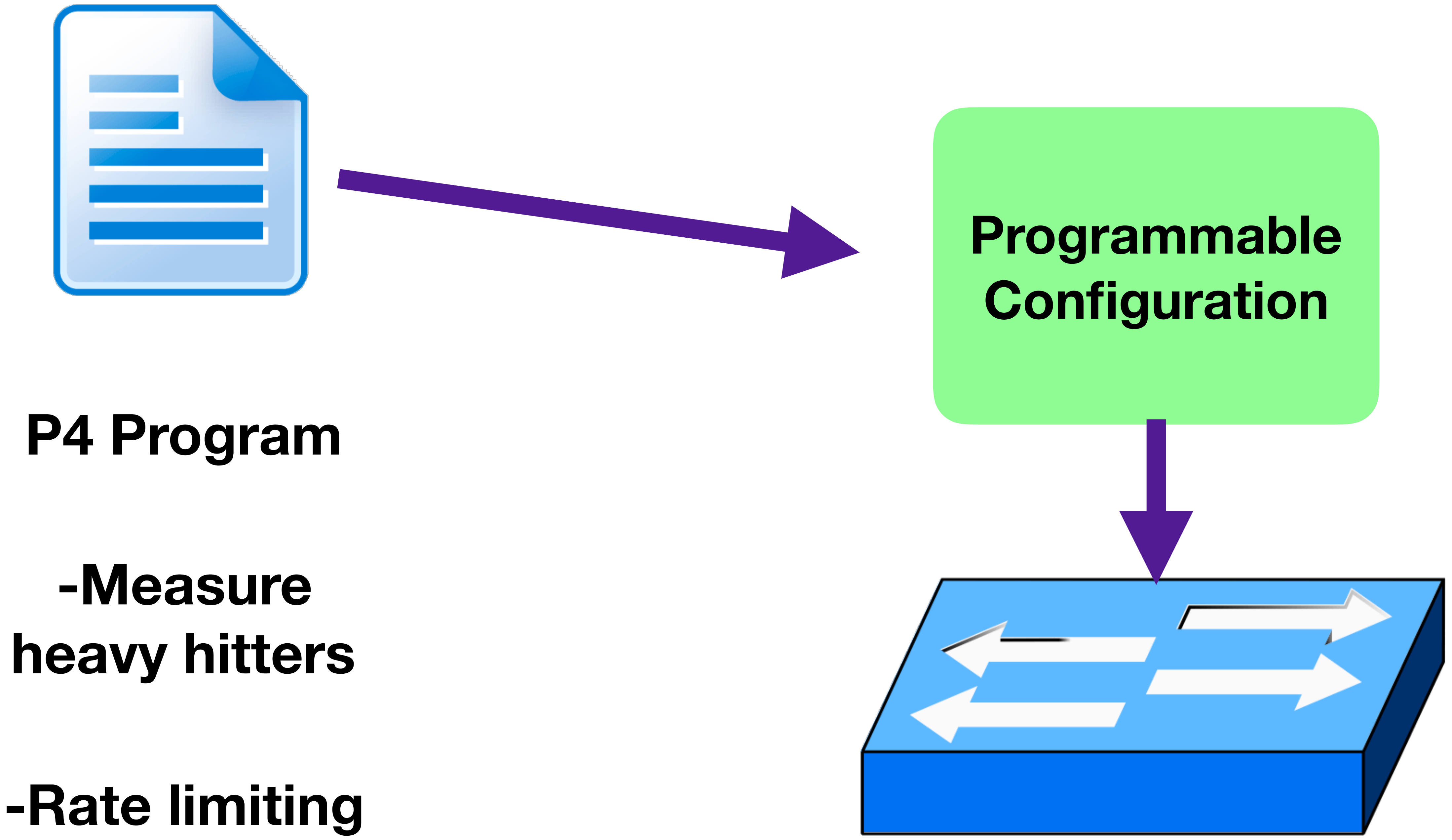


P4 Program

Programmable Configuration

PISA switch

Programming Protocol Independent Packet Processors



Programmable Configuration

P4 Program

-Measure heavy hitters

-Rate limiting

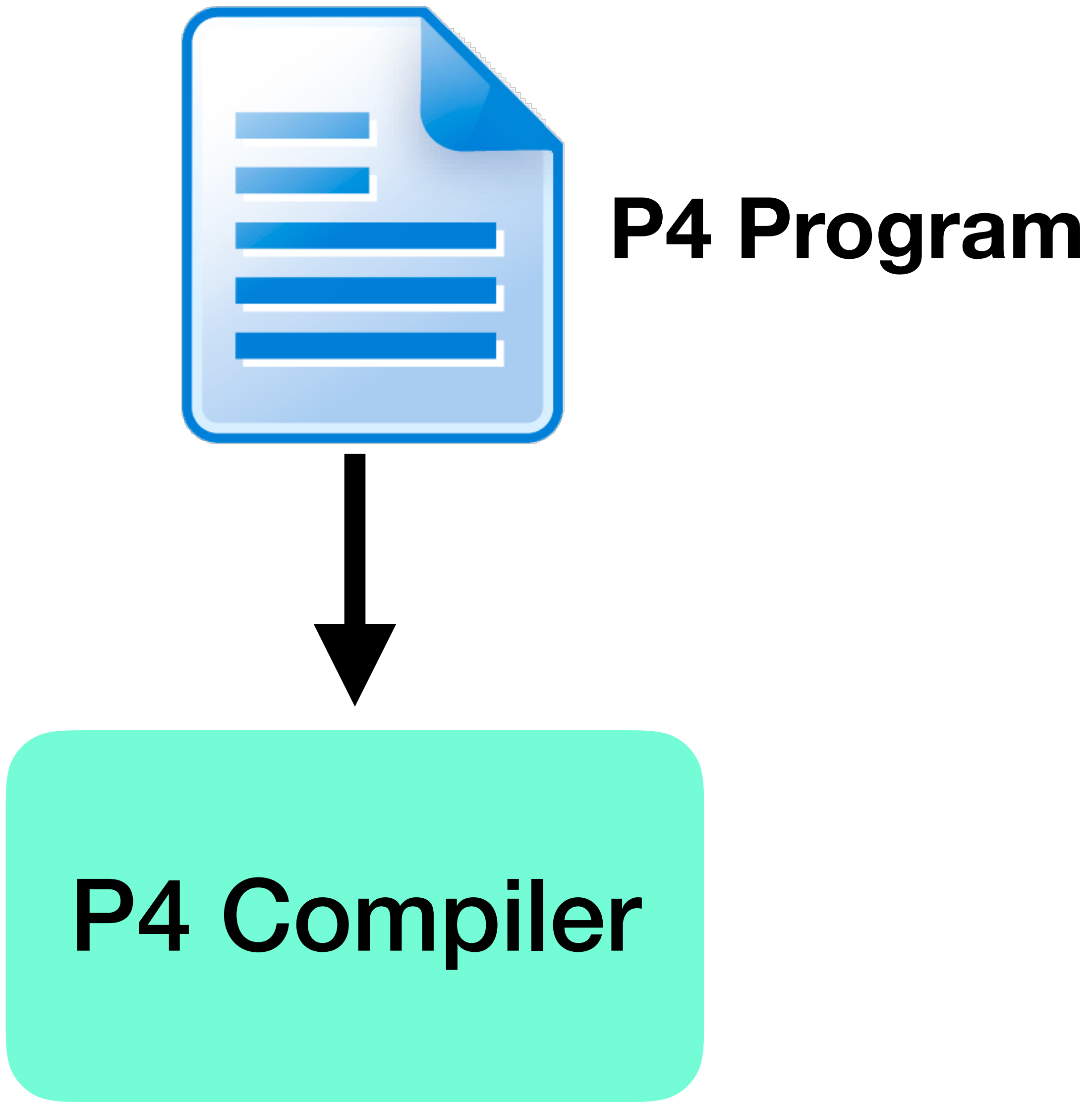
-Identify and mitigate attacks

PISA switch

P4 code should be reusable

P4 Program

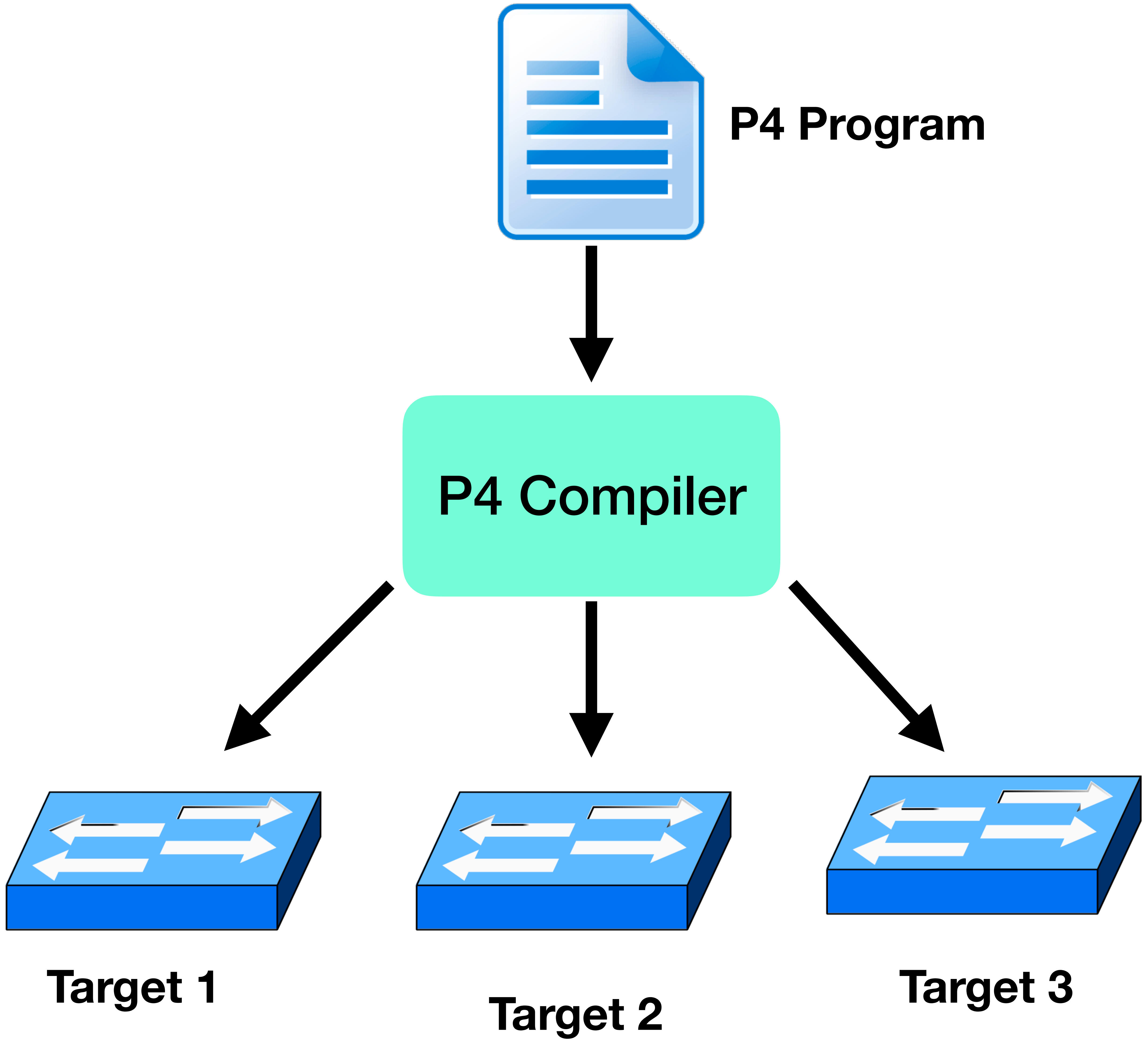
P4 code should be reusable



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

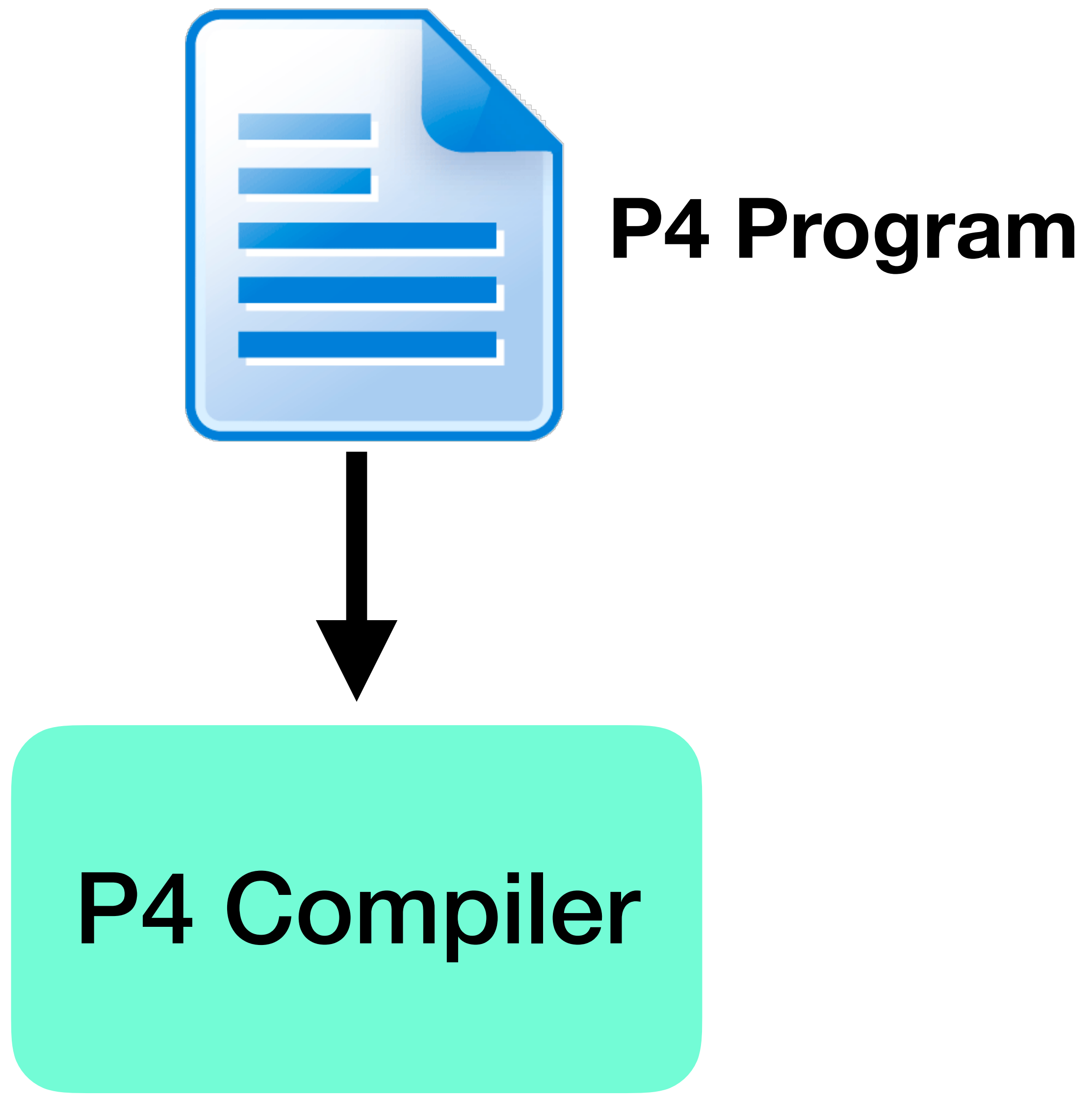
P4 makes it possible to program the network, but it does not make it easy.

Commonly used data structures are rewritten often

Circular Development

P4 Program

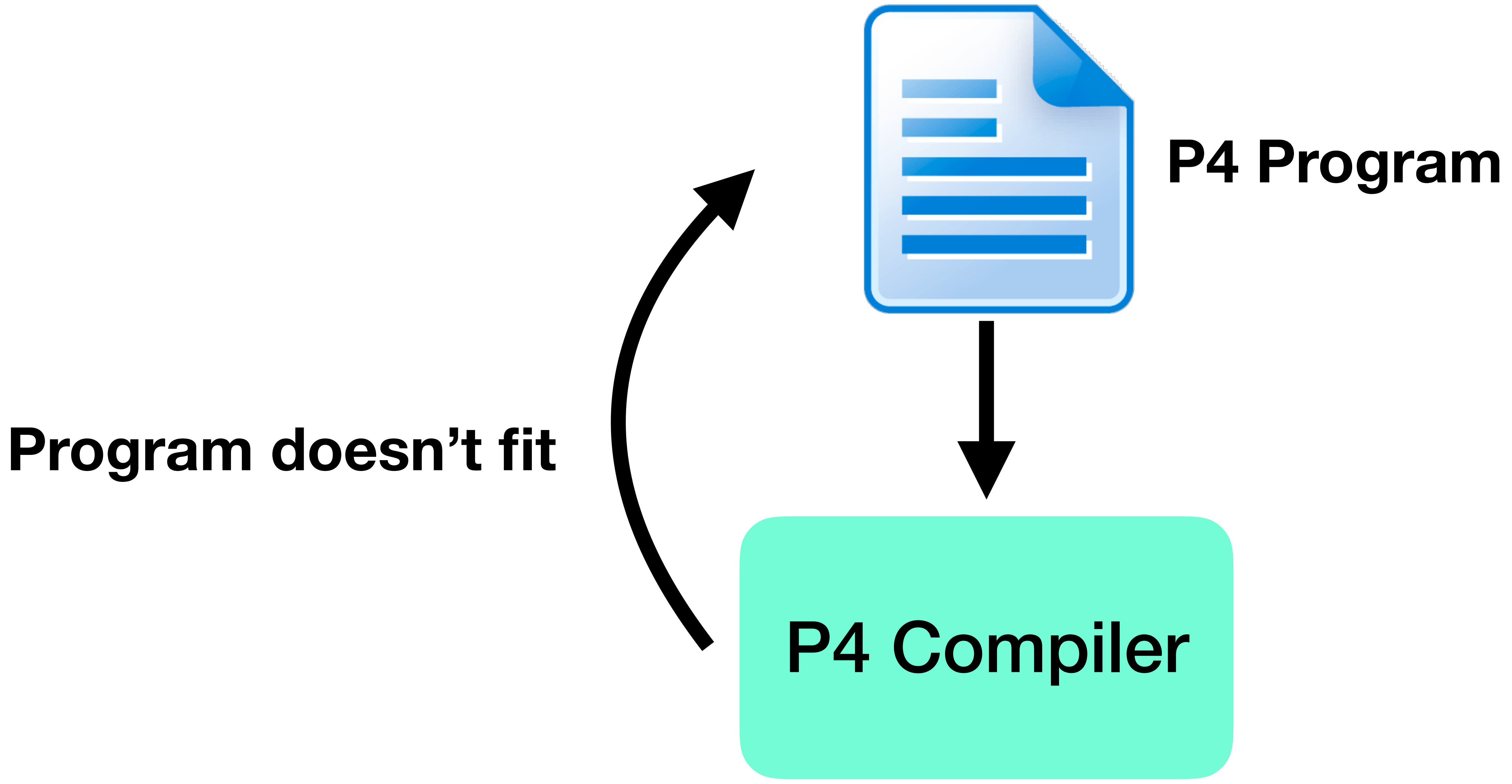
Circular Development



P4 Program

P4 Compiler

Circular Development

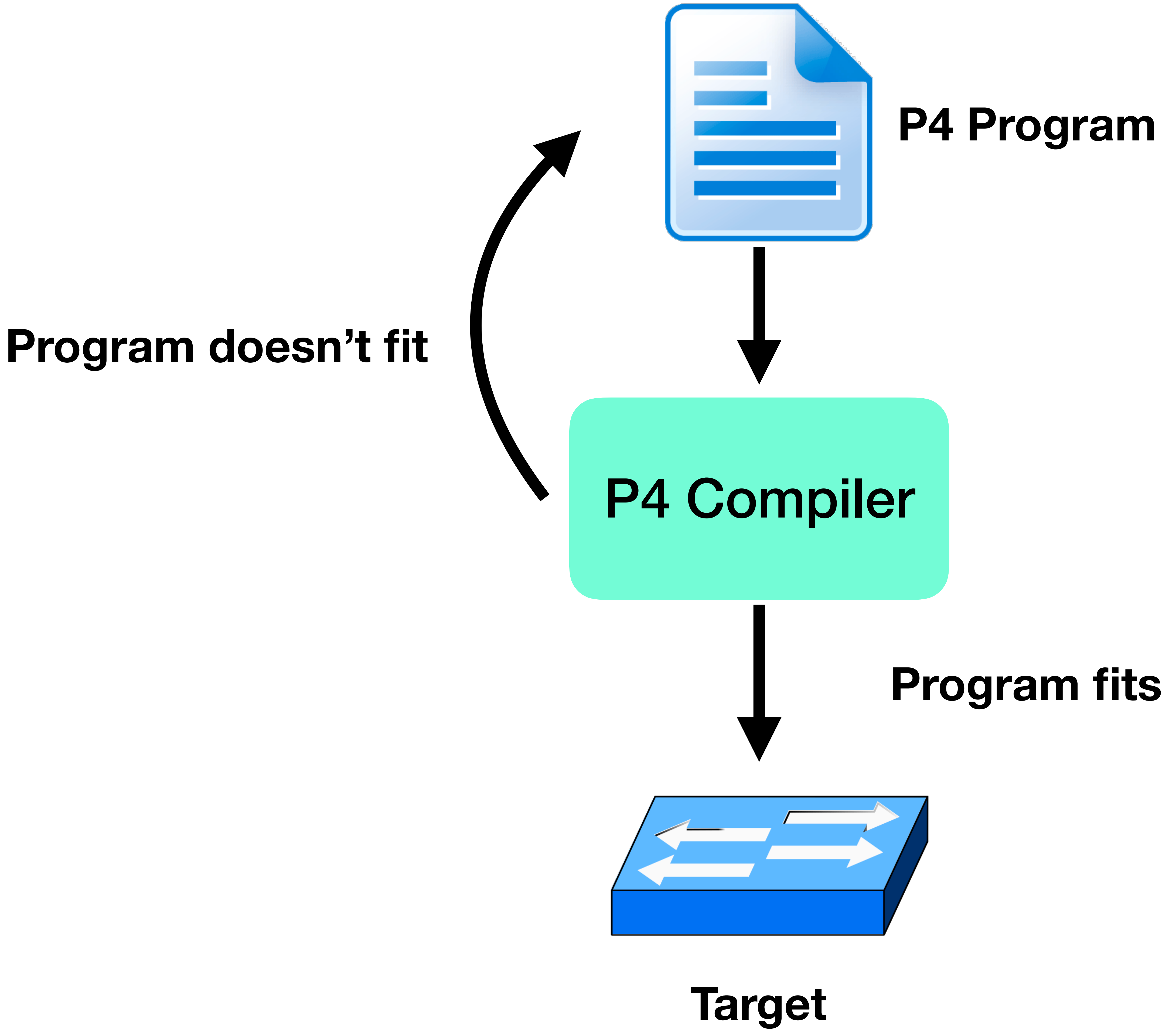


P4 Program

Program doesn’t fit

P4 Compiler

Circular Development



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

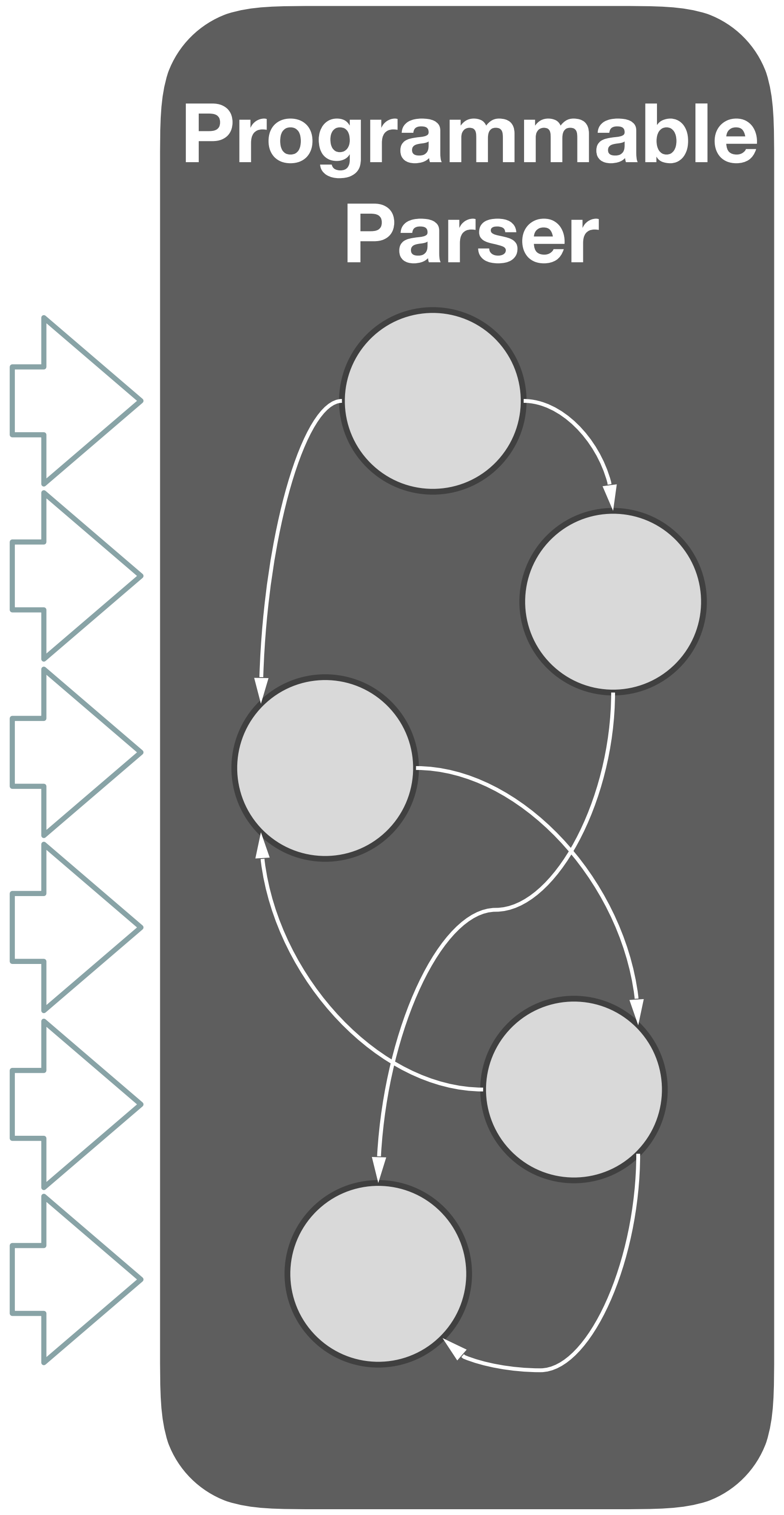
P4All

- Language
- Compiler
- Evaluation

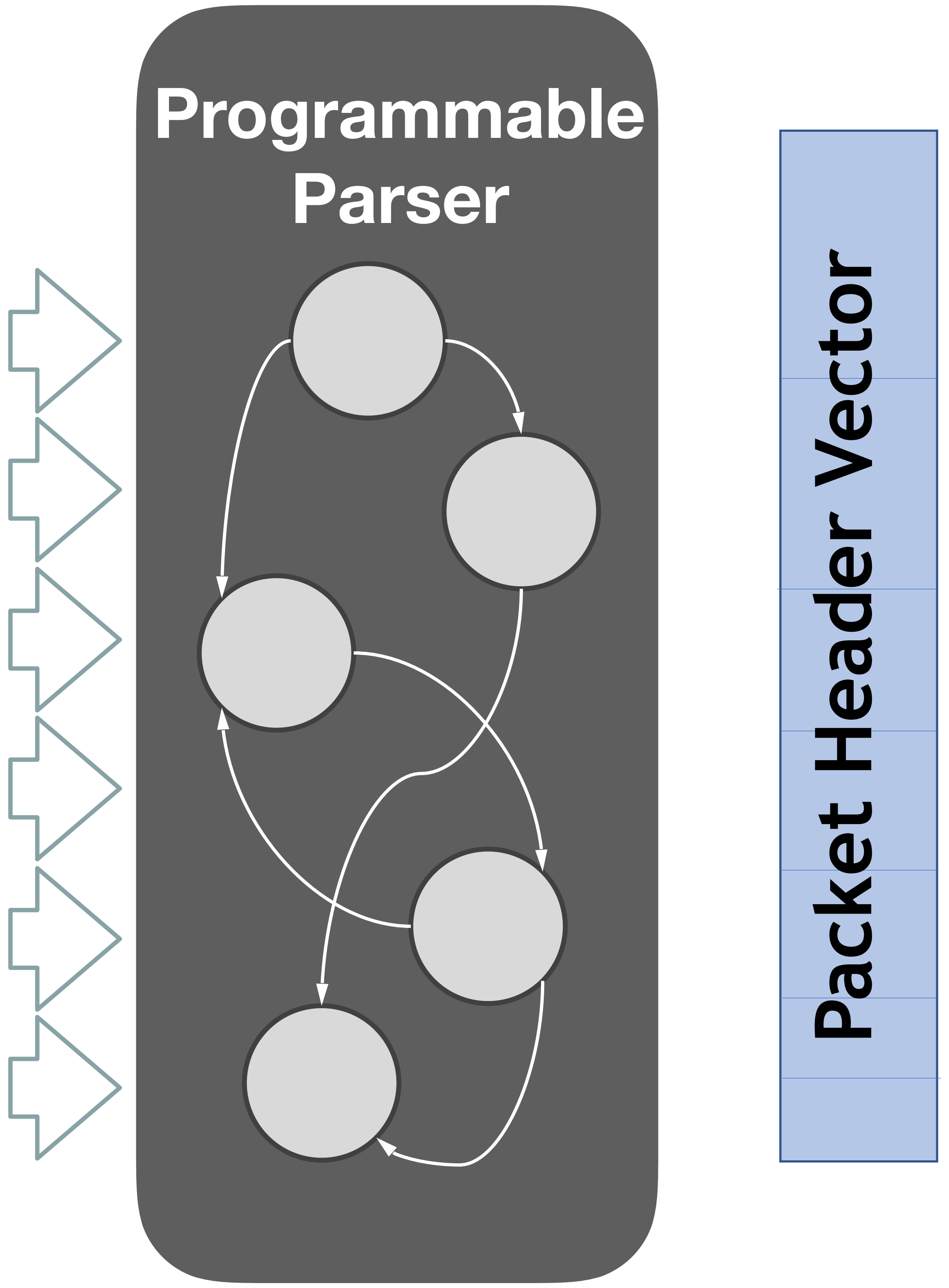
Ongoing + Future Work

Protocol-**I**ndependent **S**witch **A**rchitecture

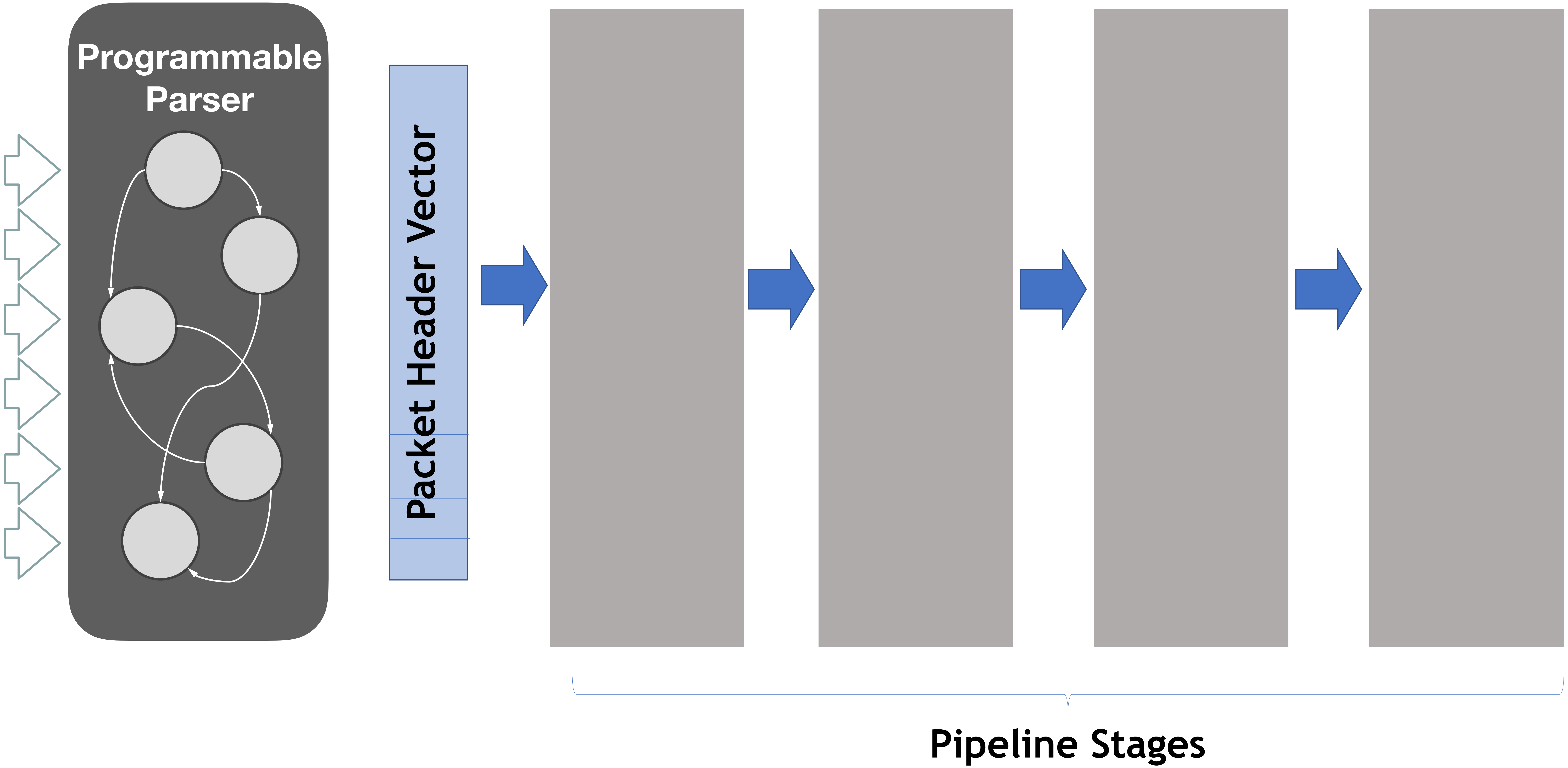
PISA



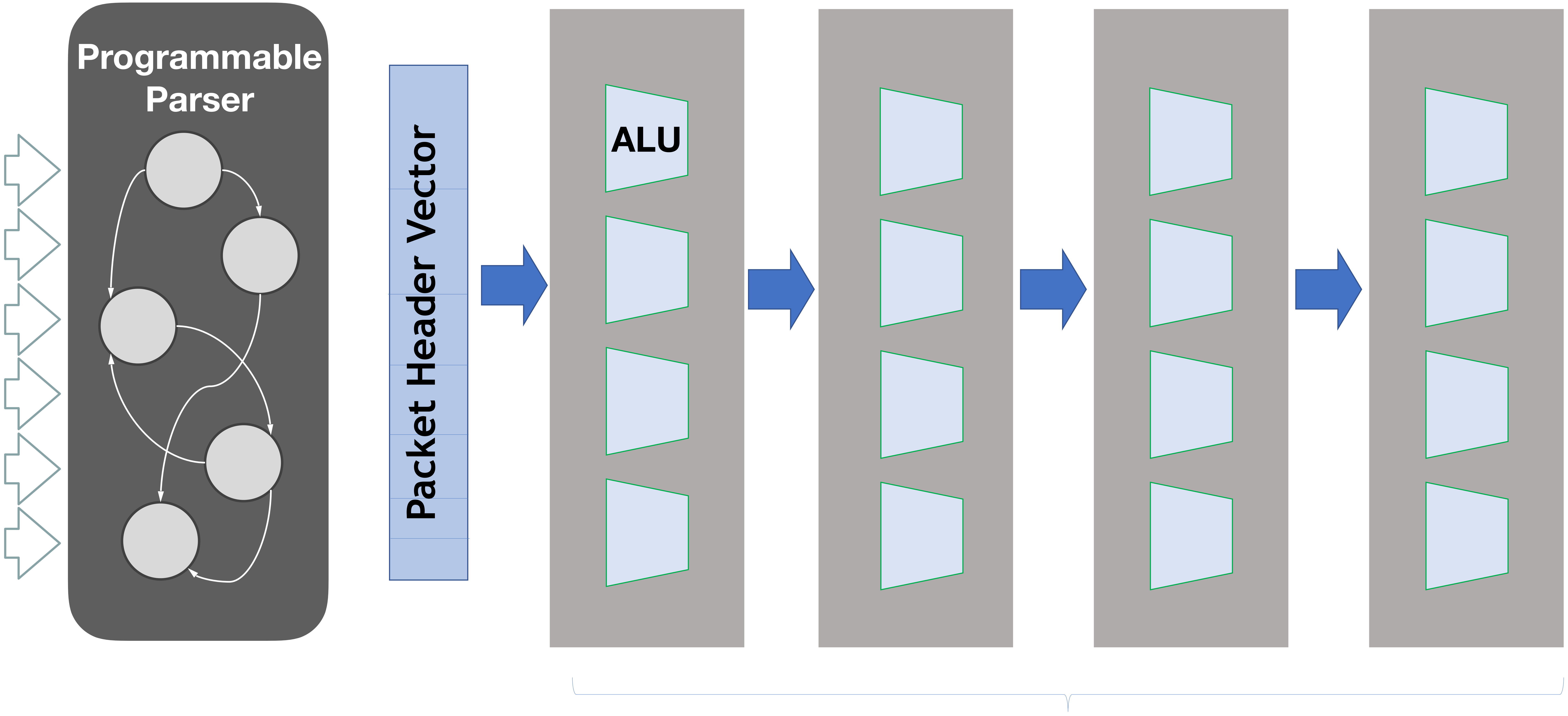
PISA



PISA

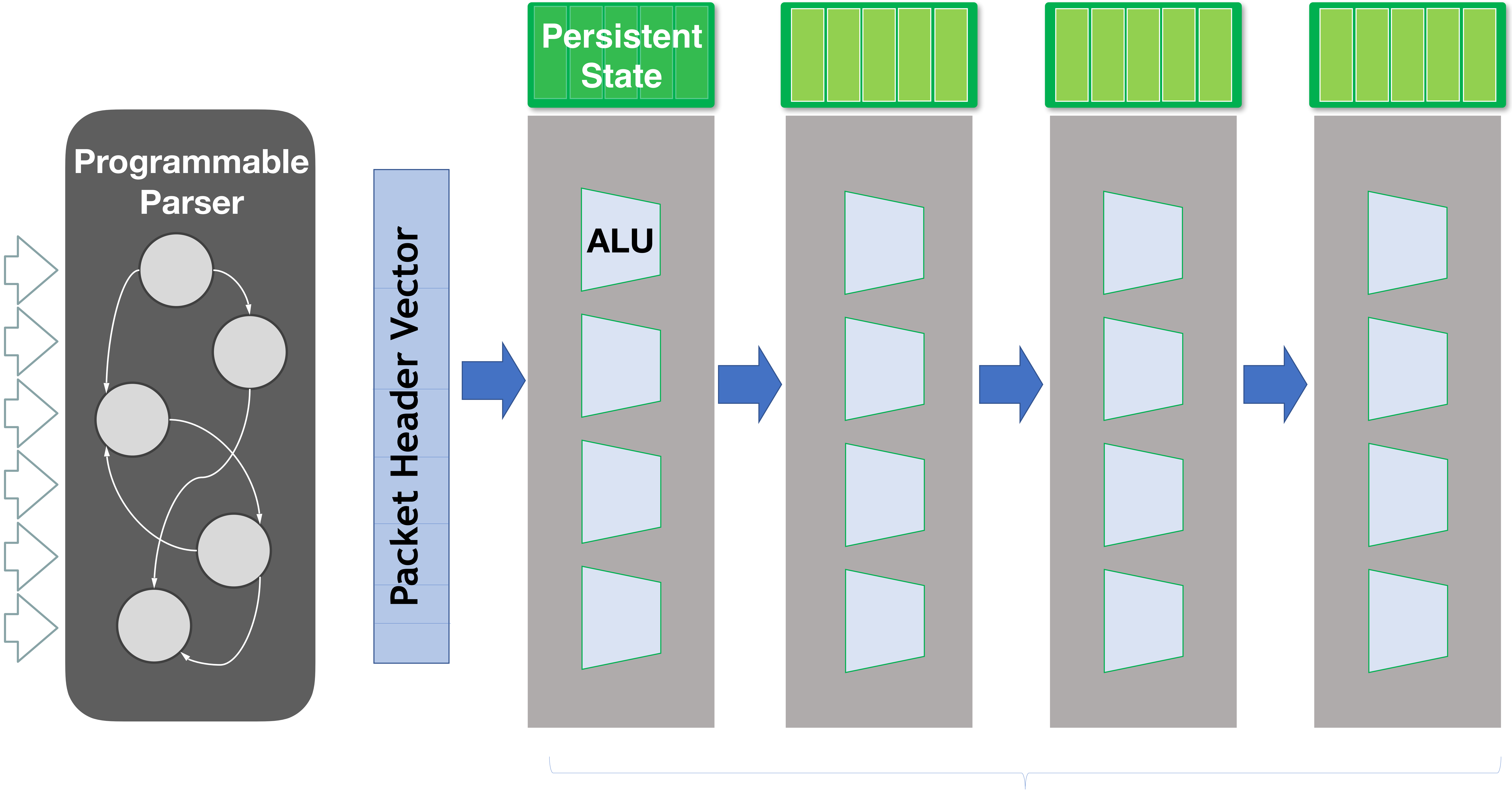

Programmable Parser
Packet Header Vector
Pipeline Stages

PISA



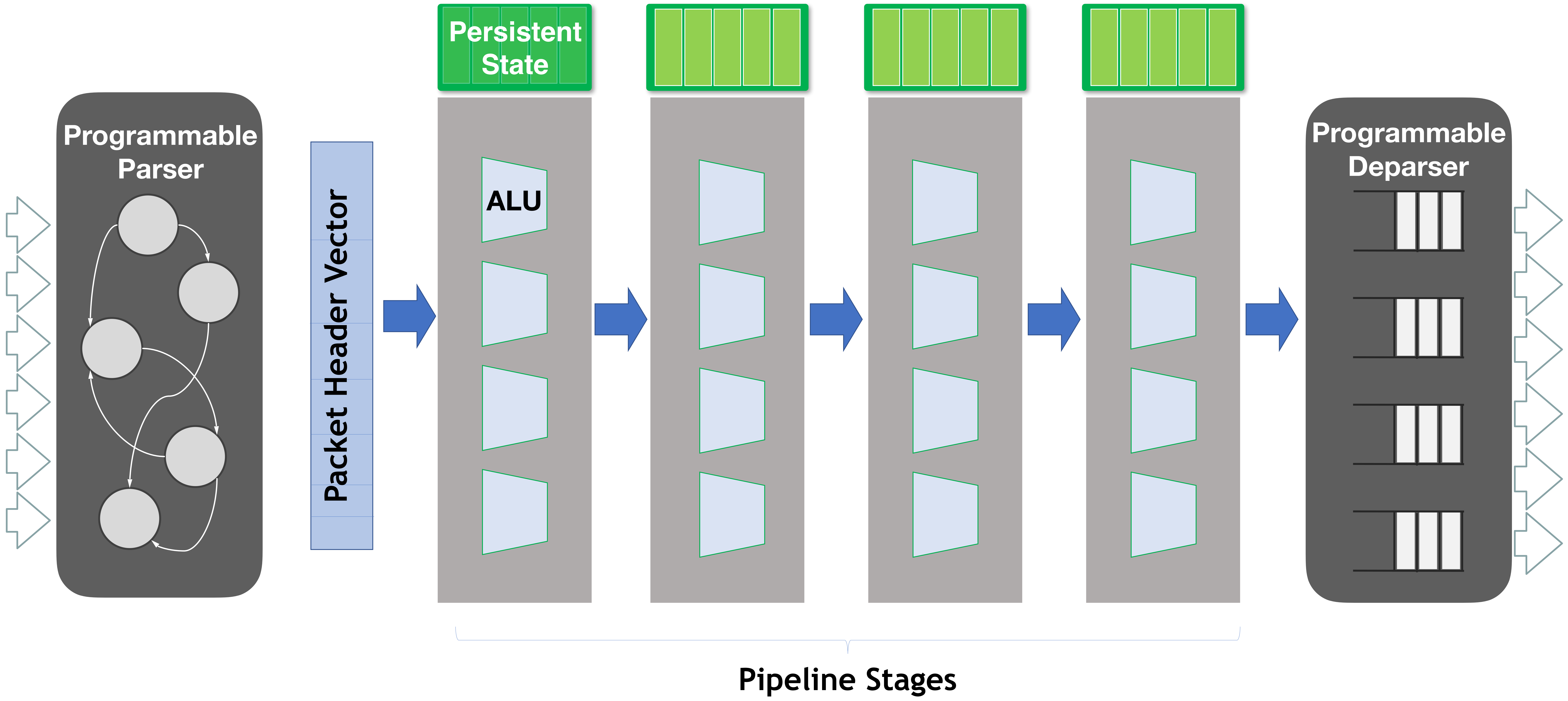
Pipeline Stages

PISA

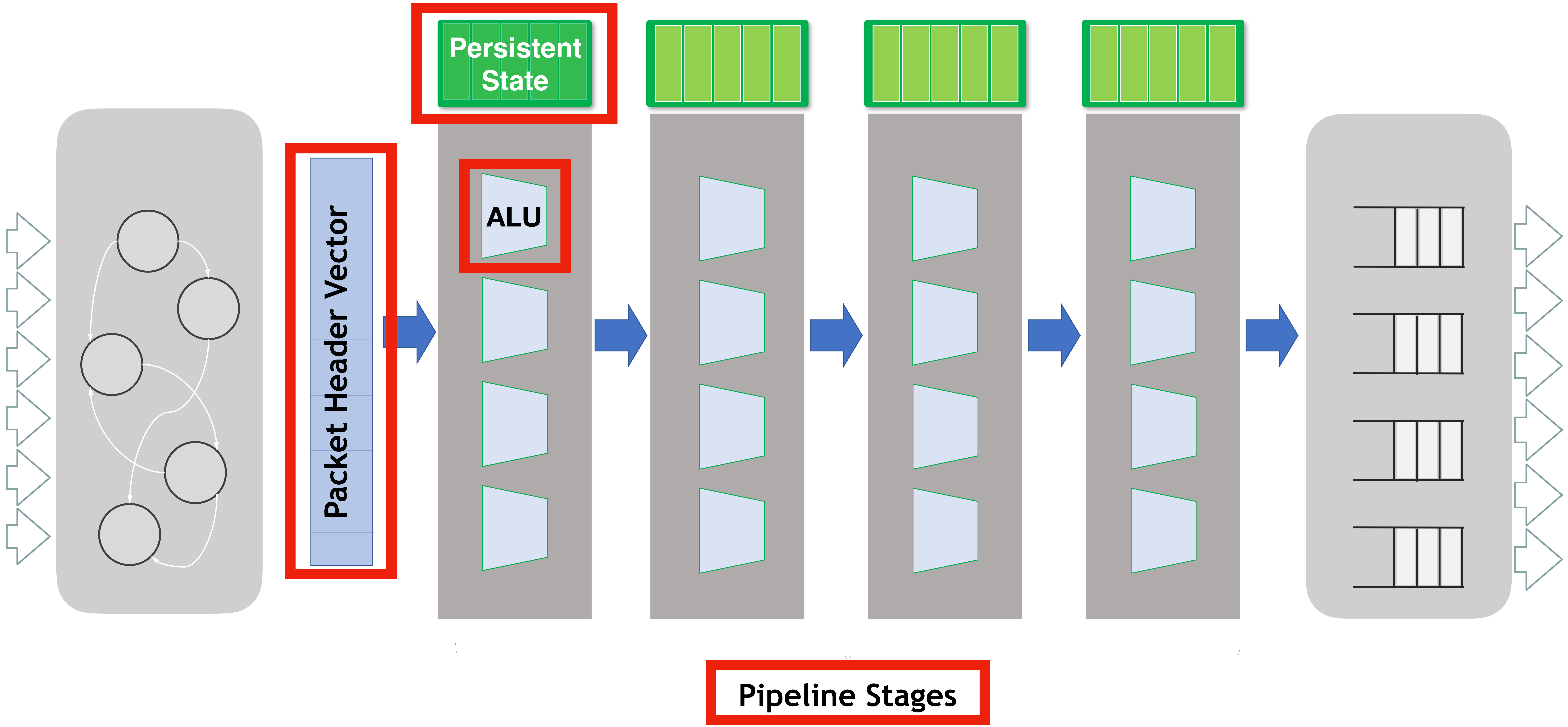


Pipeline Stages

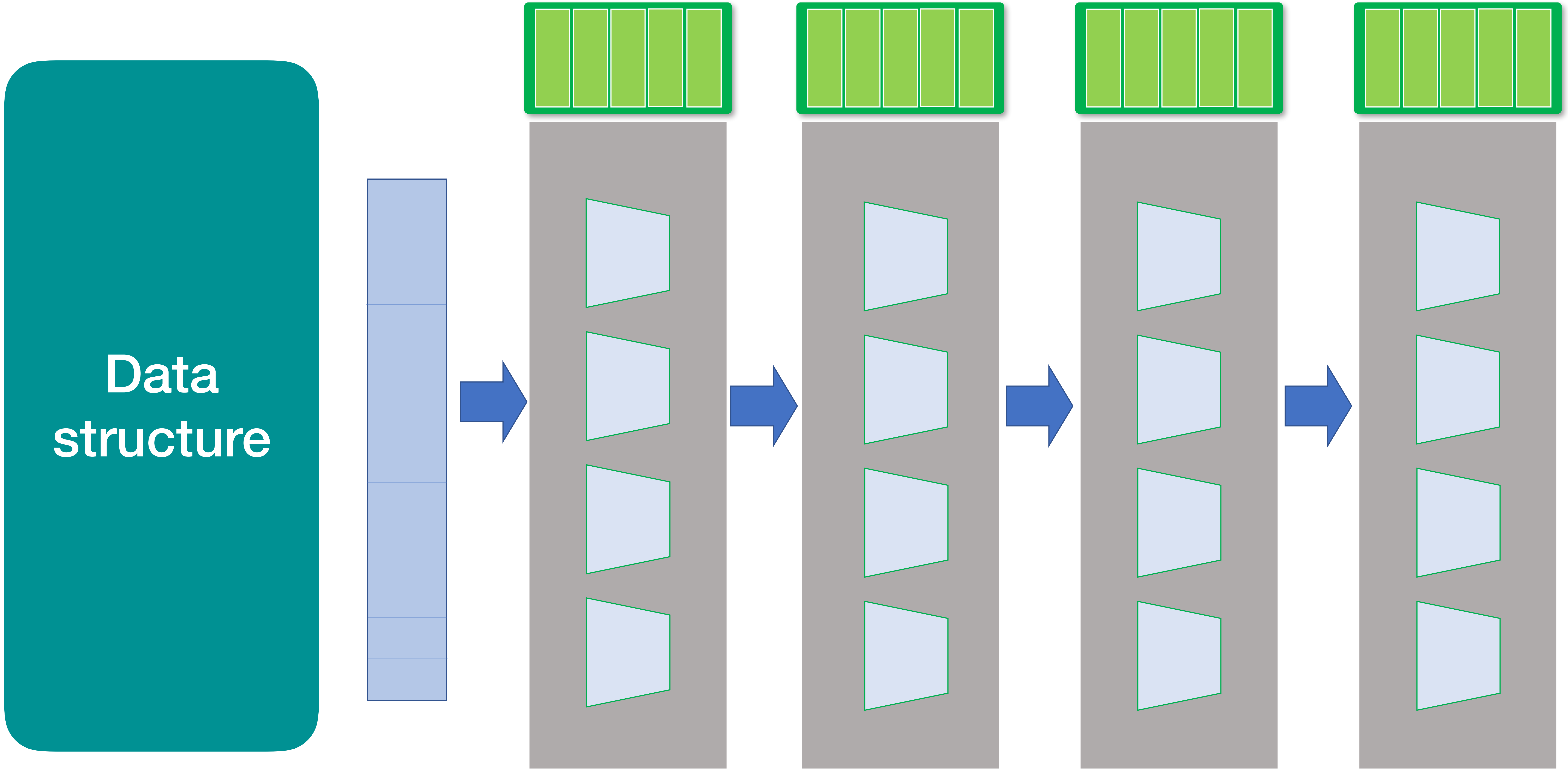
PISA


Persistent State
Programmable Parser
Packet Header Vector
ALU
Programmable Deparser
Pipeline Stages

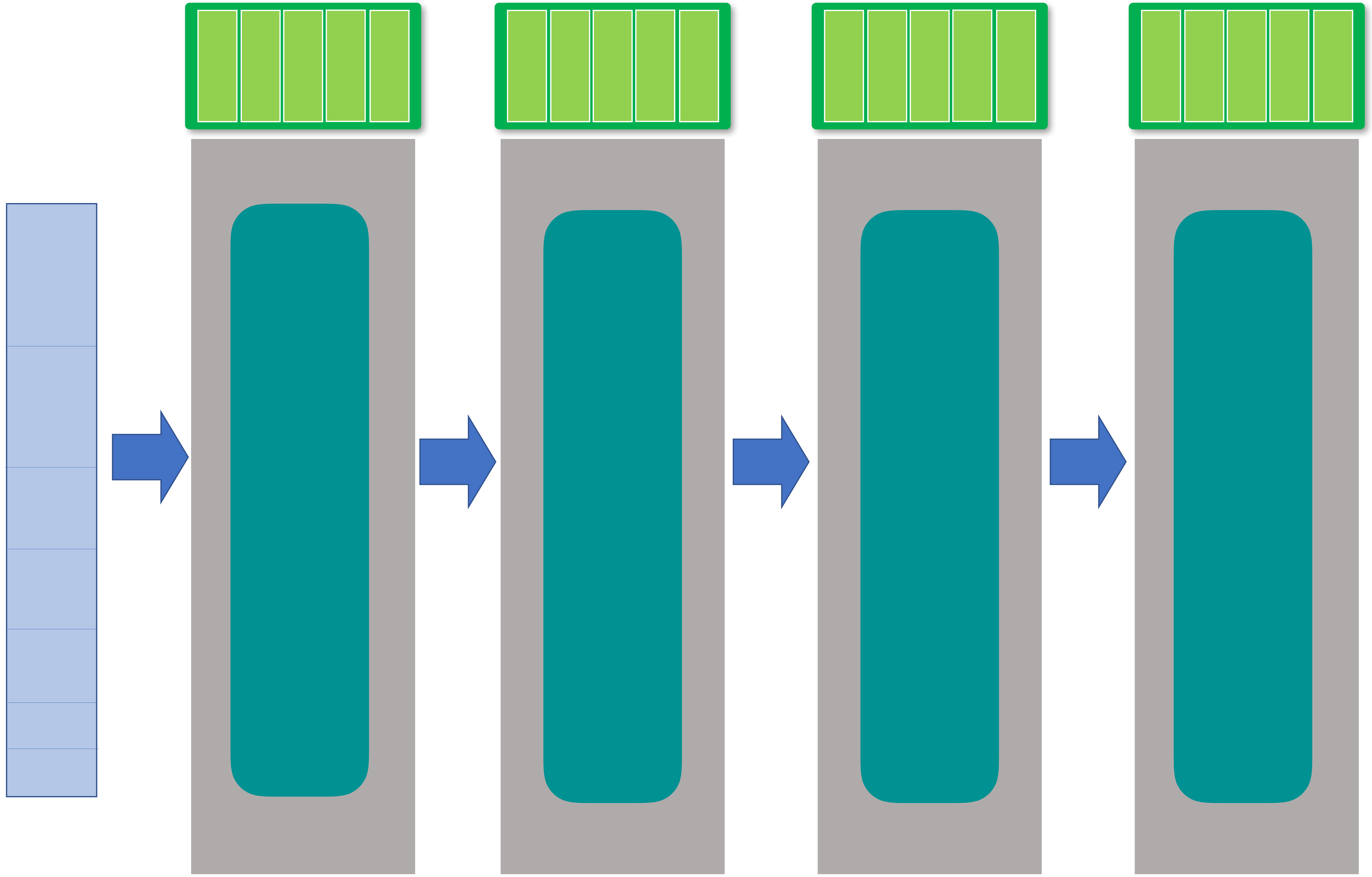
PISA


Persistent State
Packet Header Vector
ALU
Pipeline Stages

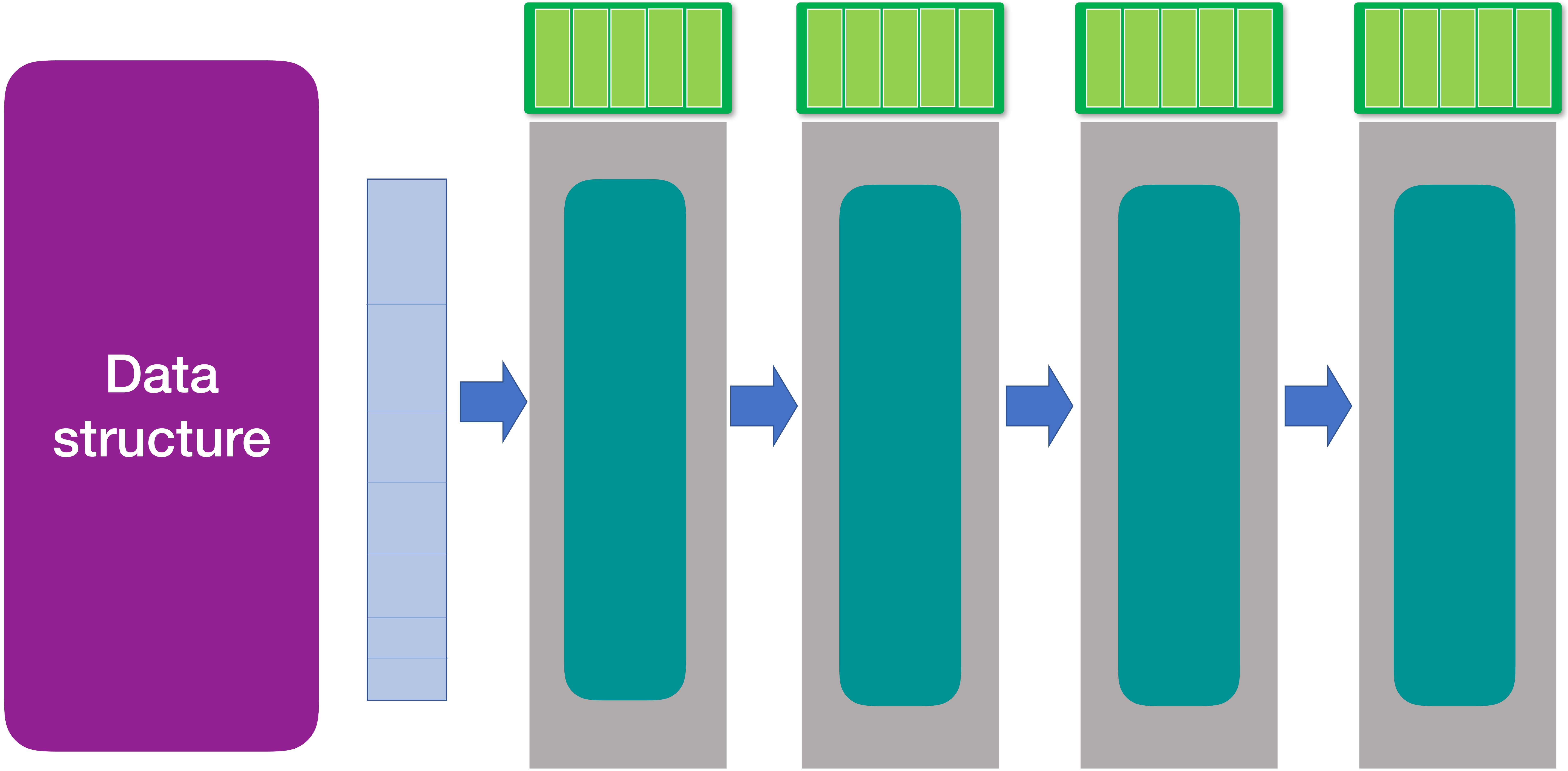
PISA



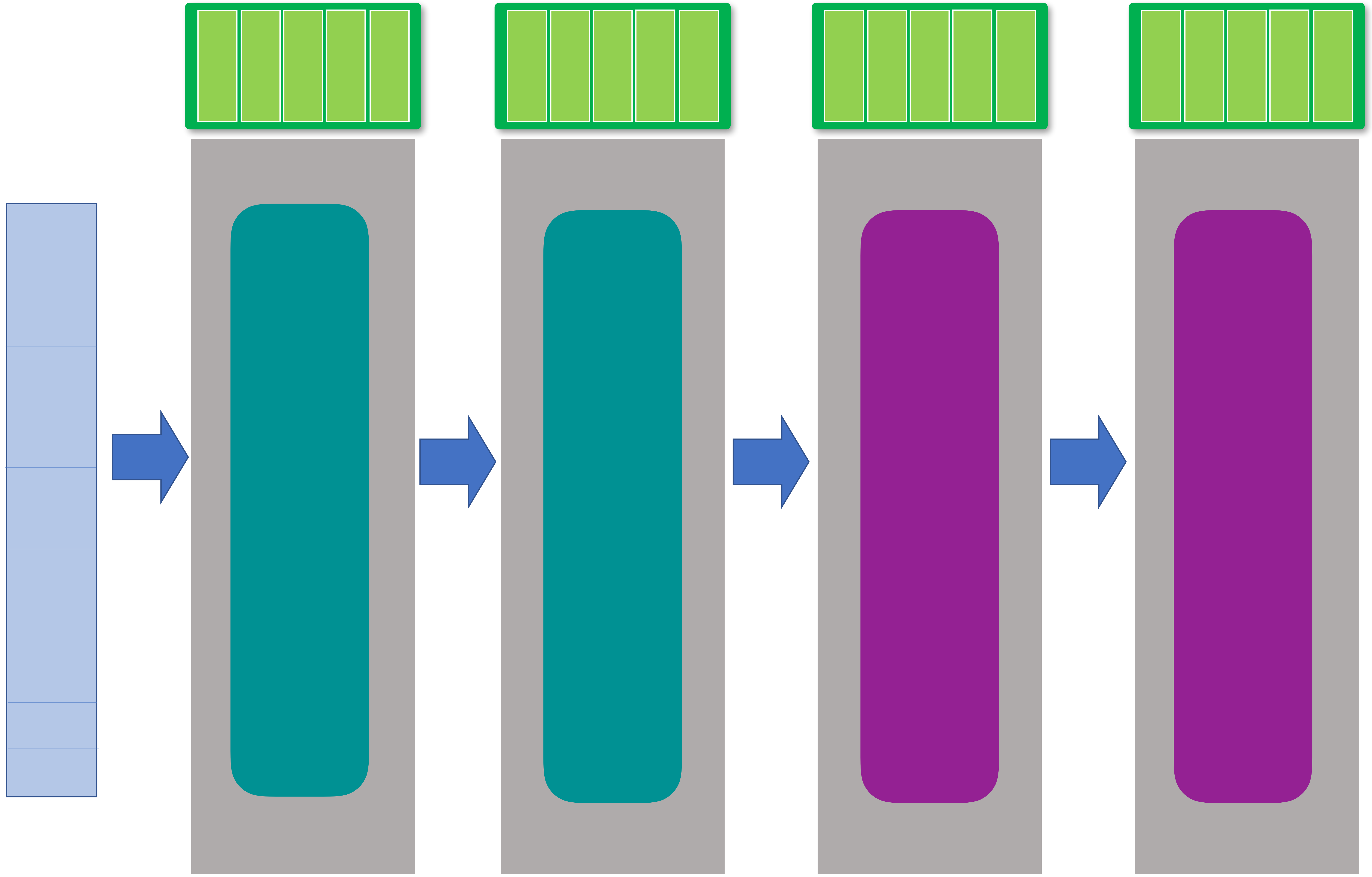
PISA



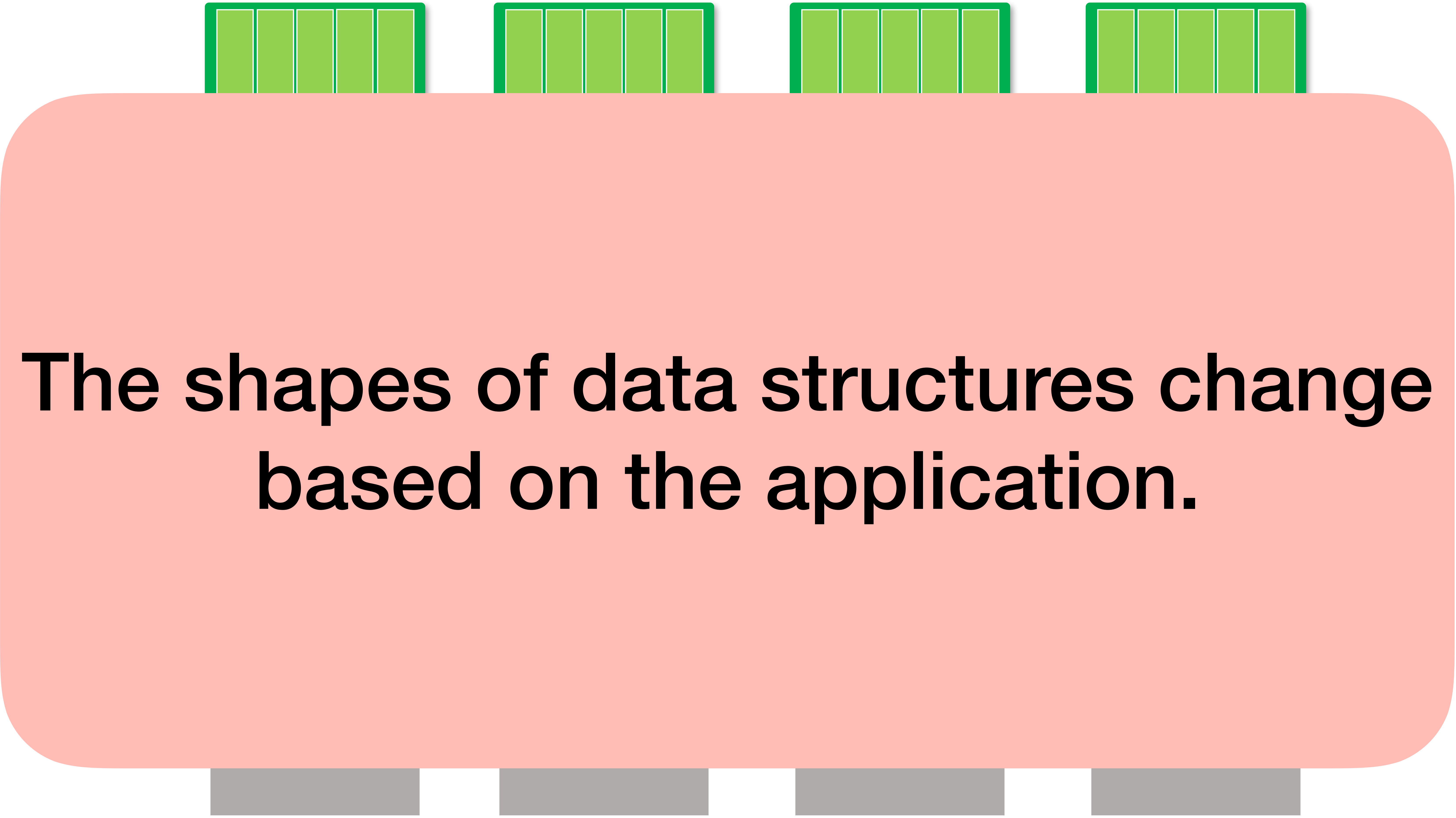
PISA



PISA



PISA

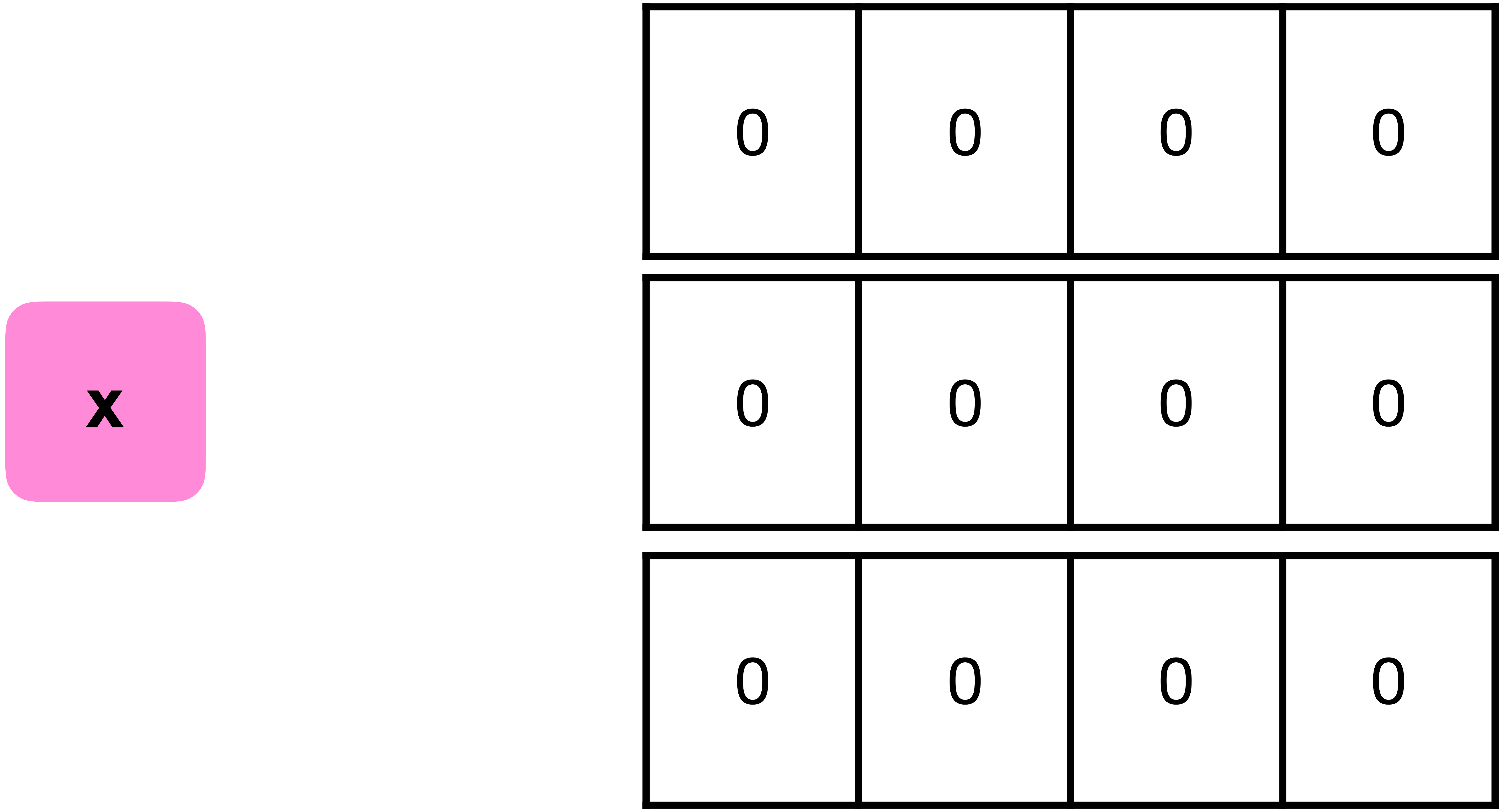


The shapes of data structures change based on the application.

Count-Min Sketch

0	0	0	0
0	0	0	0
0	0	0	0

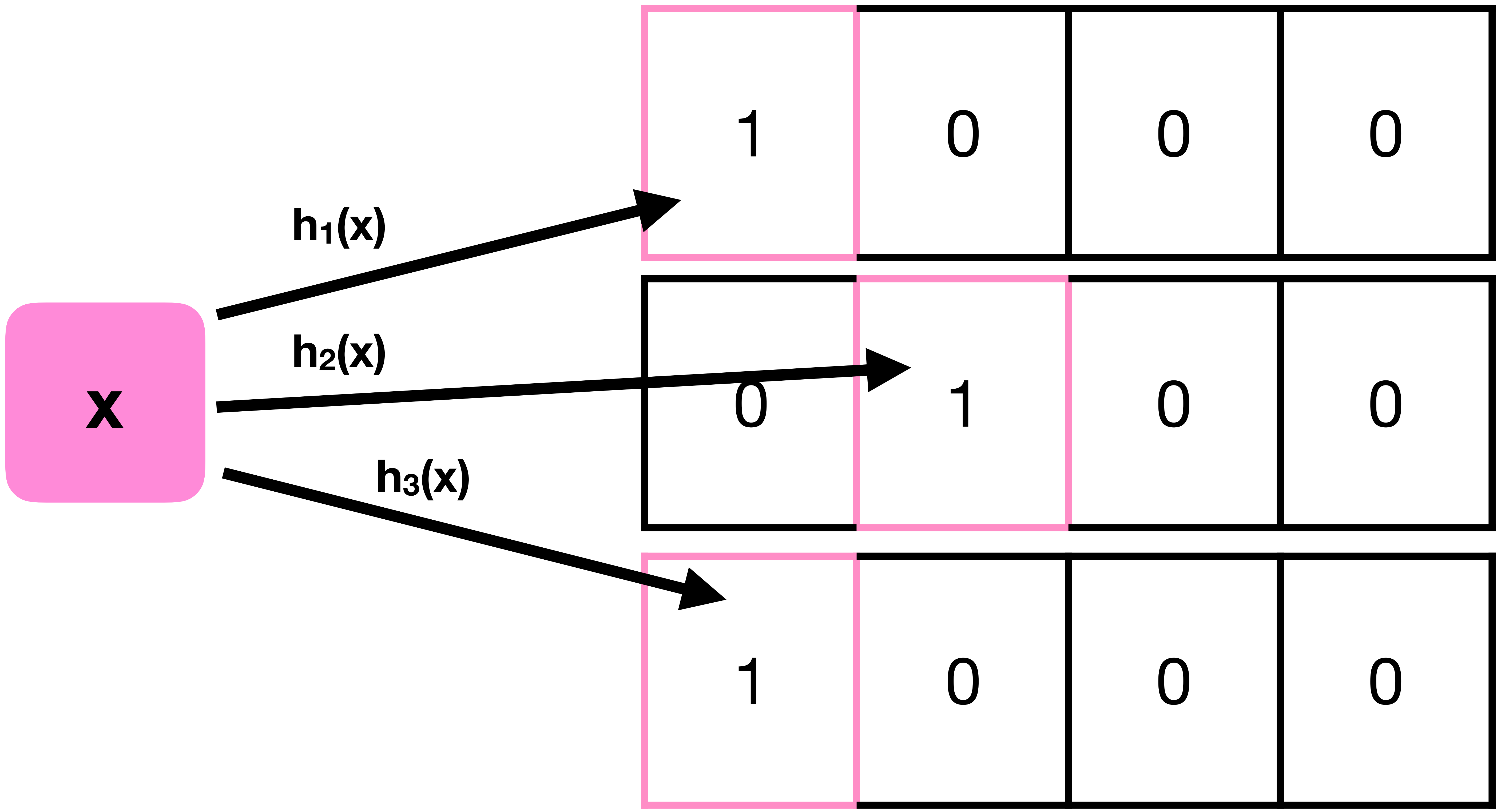
Count-Min Sketch



x

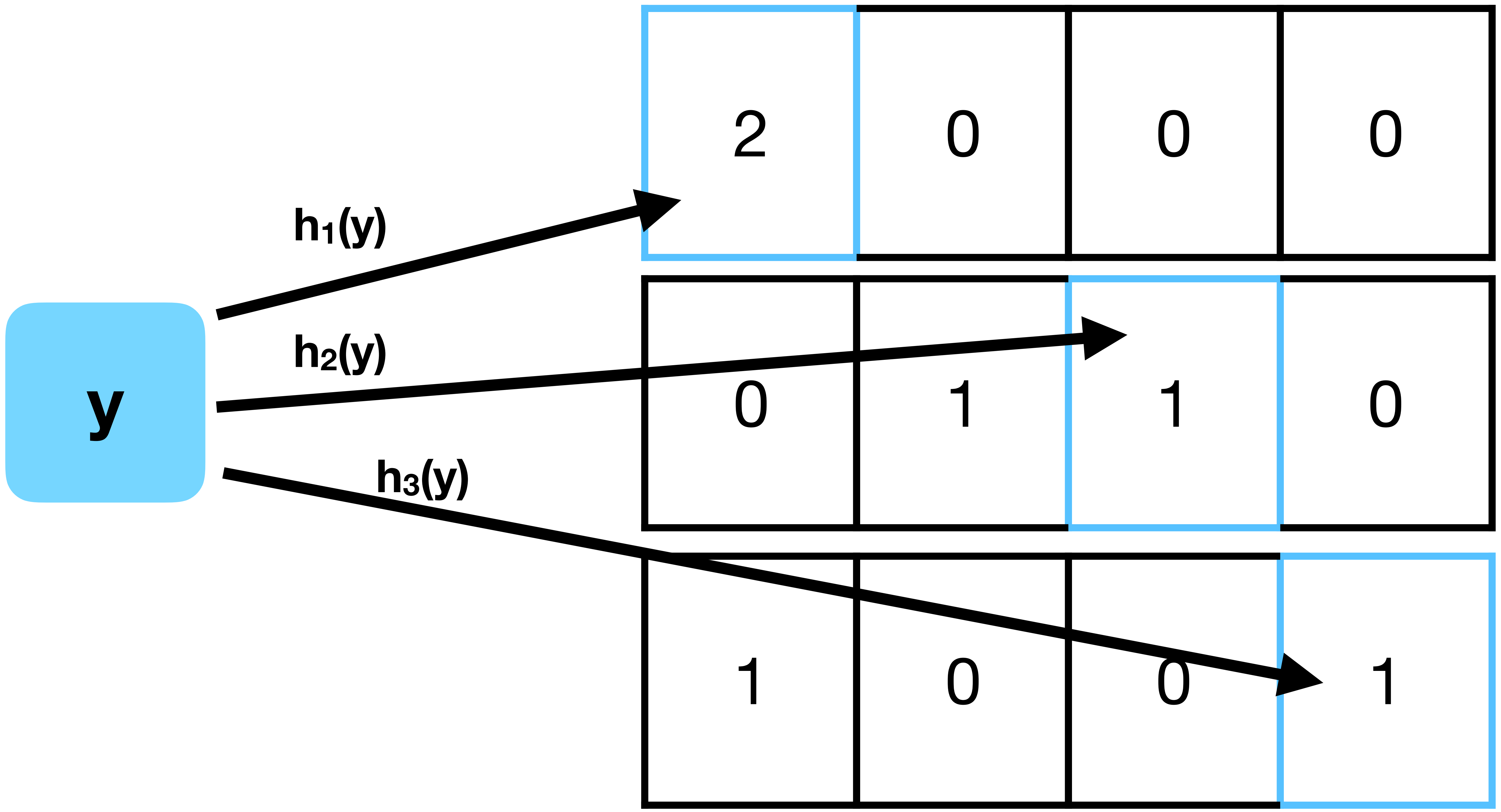
0	0	0	0
0	0	0	0
0	0	0	0

Count-Min Sketch



1	0	0	0
0	1	0	0
1	0	0	0

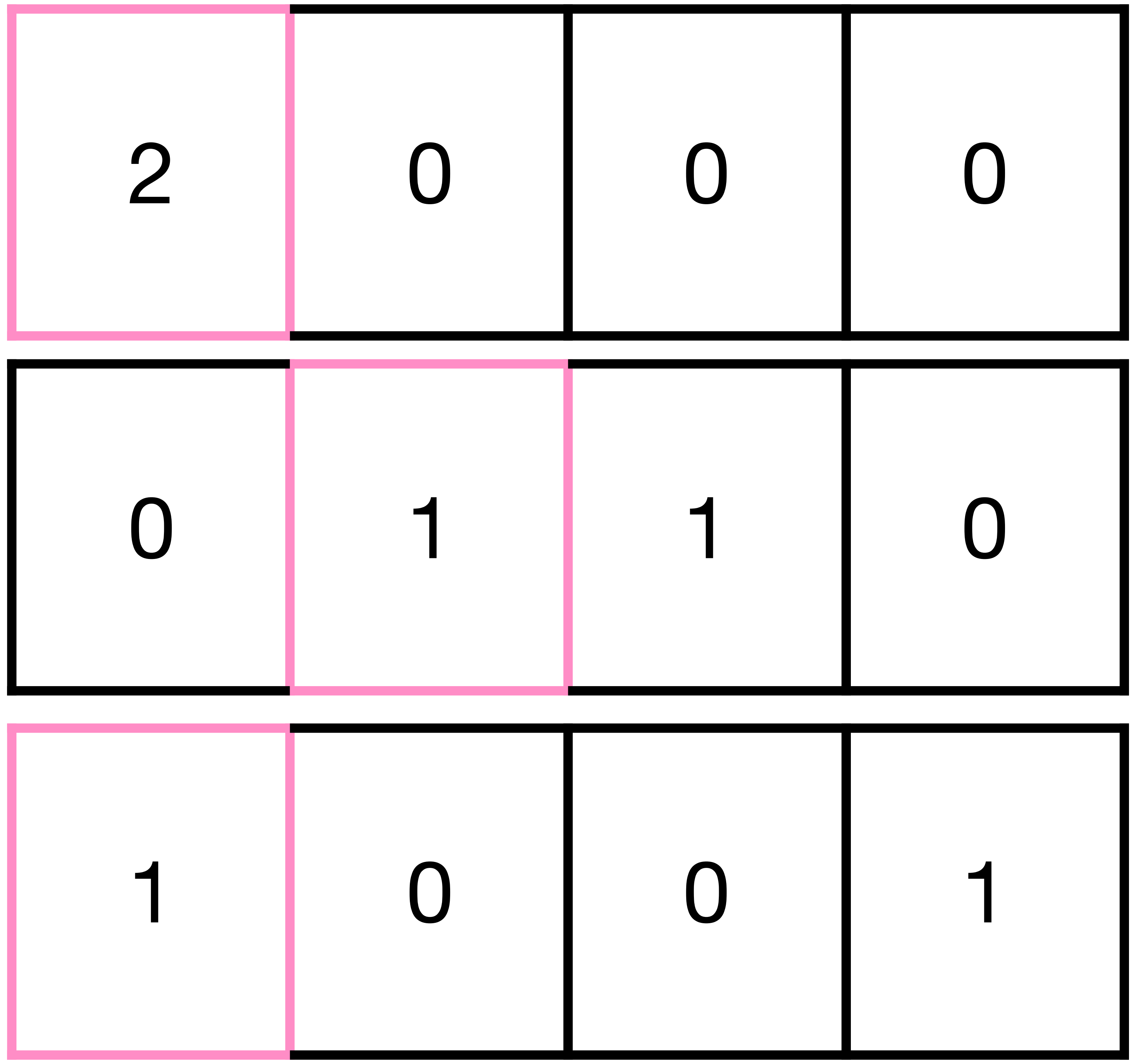
Count-Min Sketch



2	0	0	0
0	1	1	0
1	0	0	1

Count-Min Sketch

x

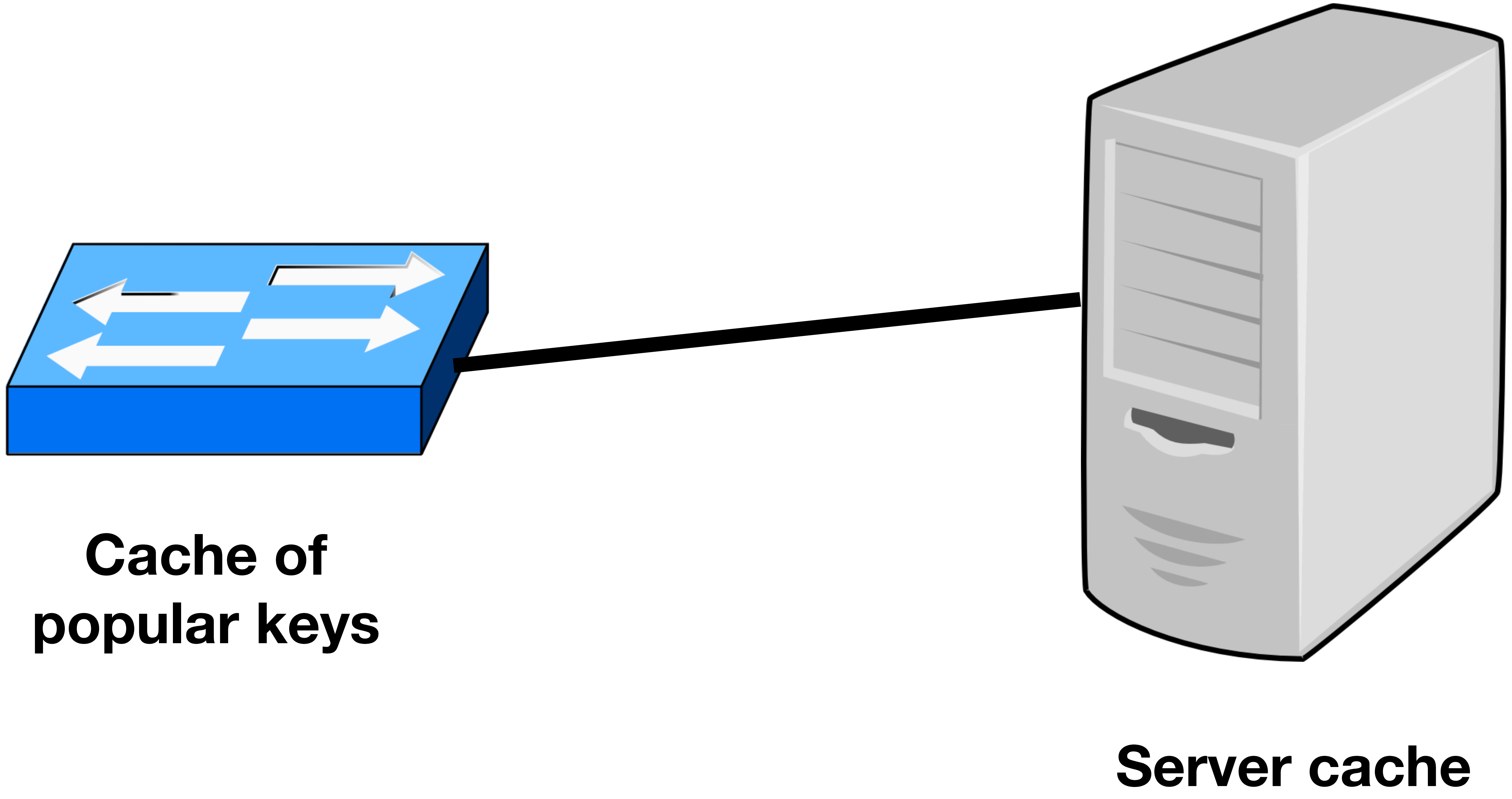


2	0	0	0
0	1	1	0
1	0	0	1

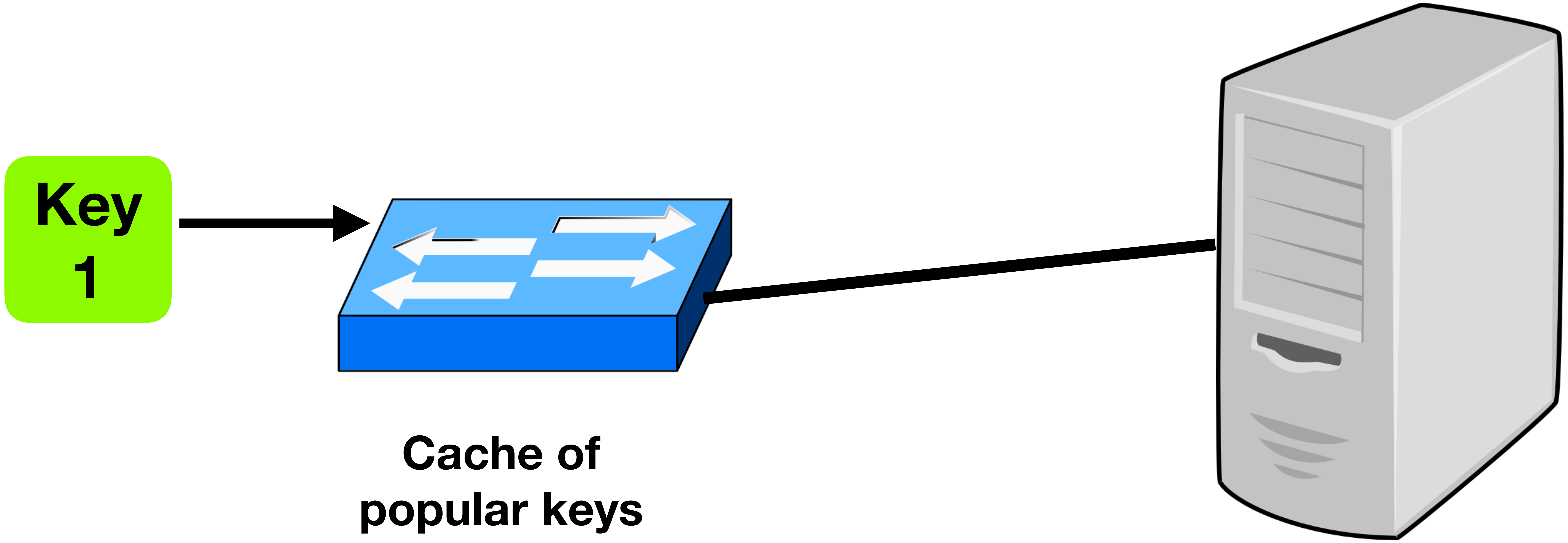
Count(x) = 1

Data Plane Caching

Data Plane Caching



Data Plane Caching

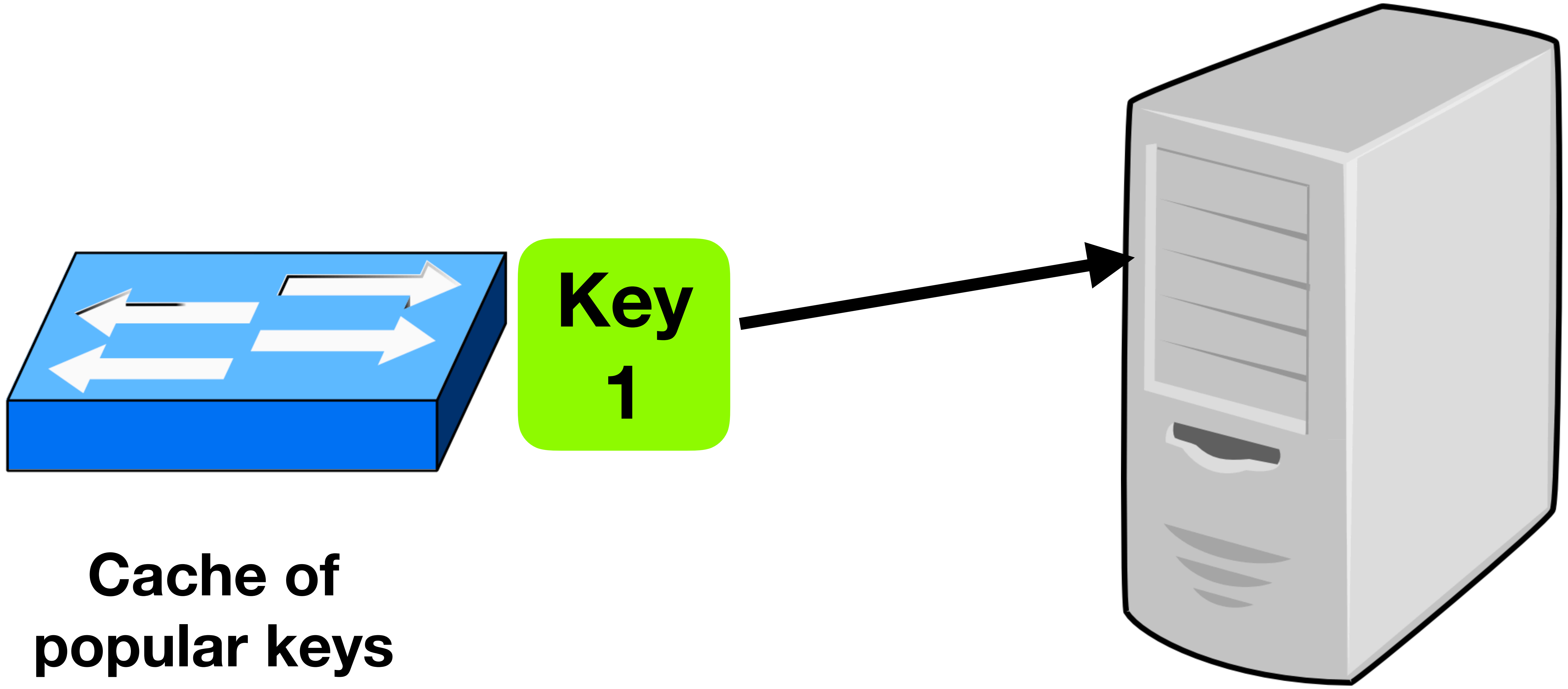


Key 1

Cache of popular keys

Server cache

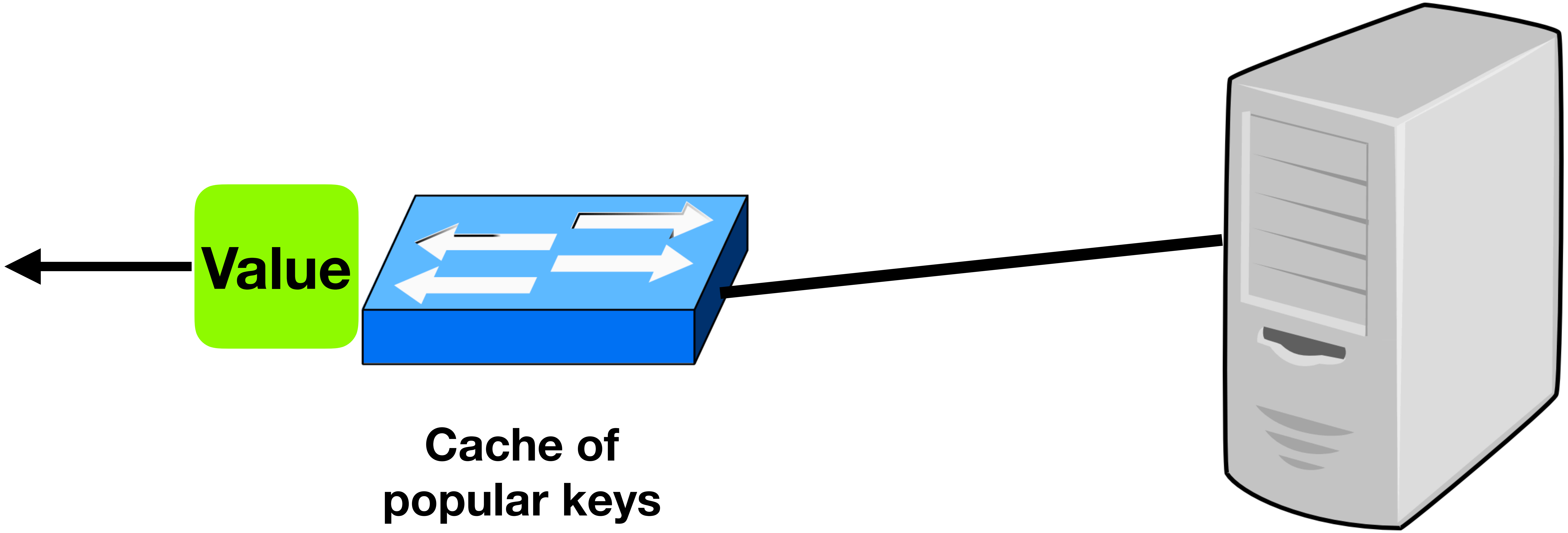
Data Plane Caching



Cache of popular keys

Server cache

Data Plane Caching



Value

Cache of
popular keys

Server cache

Data Plane Caching

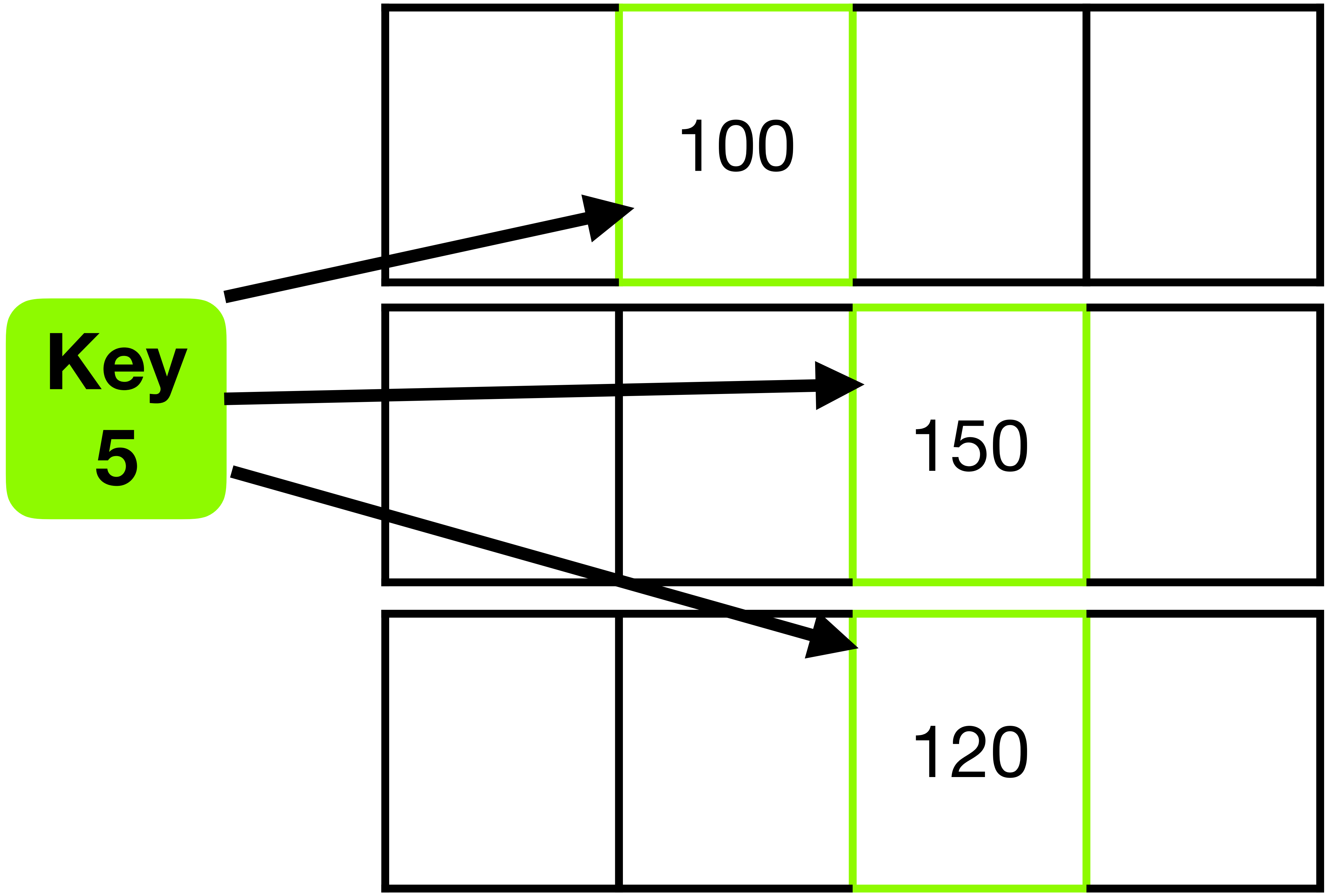
Key	Value
1	A
2	B
3	C
4	D

Cache of popular keys

Tracking Key Popularity

Key	Value
1	A
2	B
3	C
4	D

Cache of popular keys



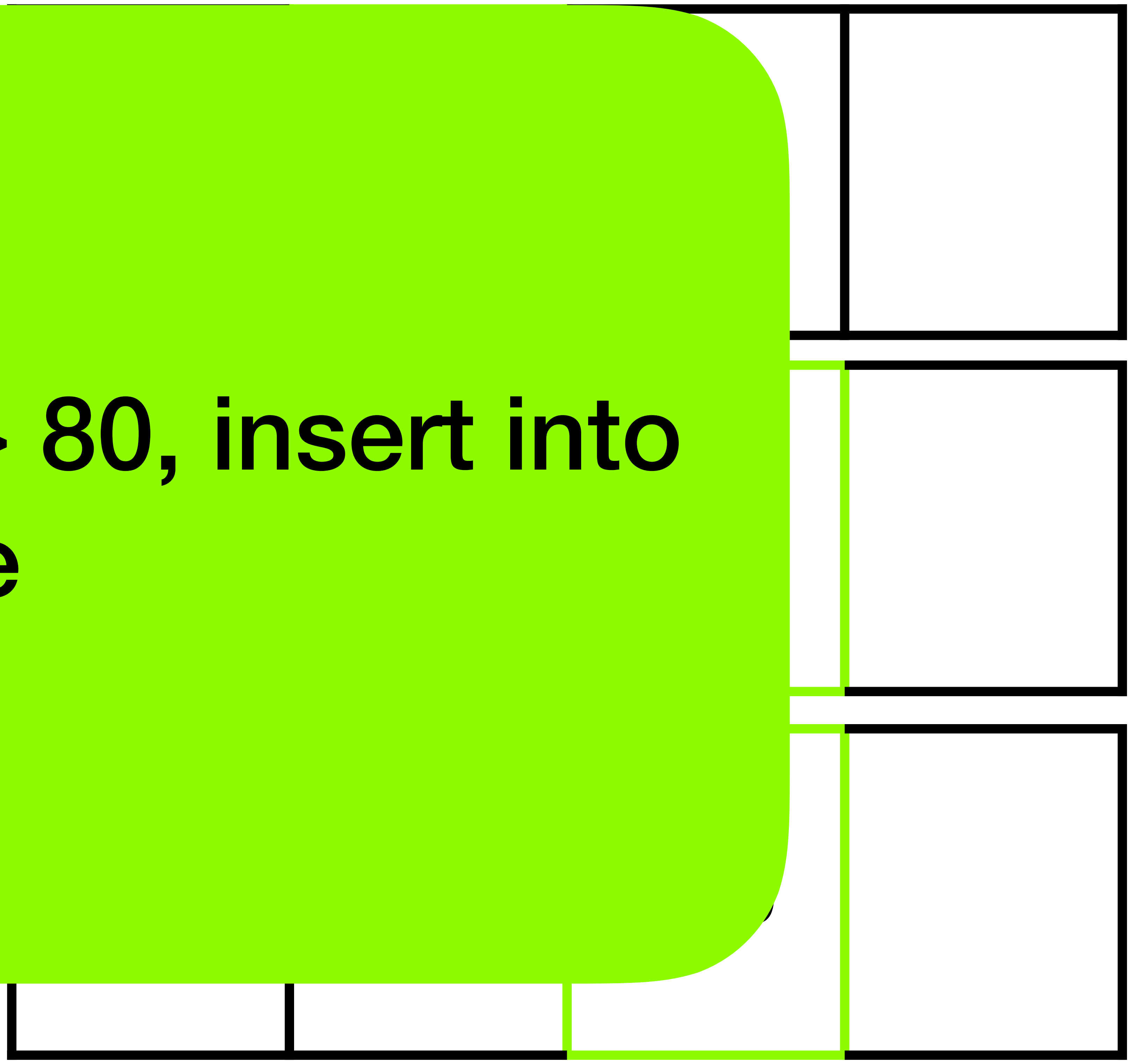
CMS

Tracking Key Popularity

Key	Value
1	
2	
3	
4	

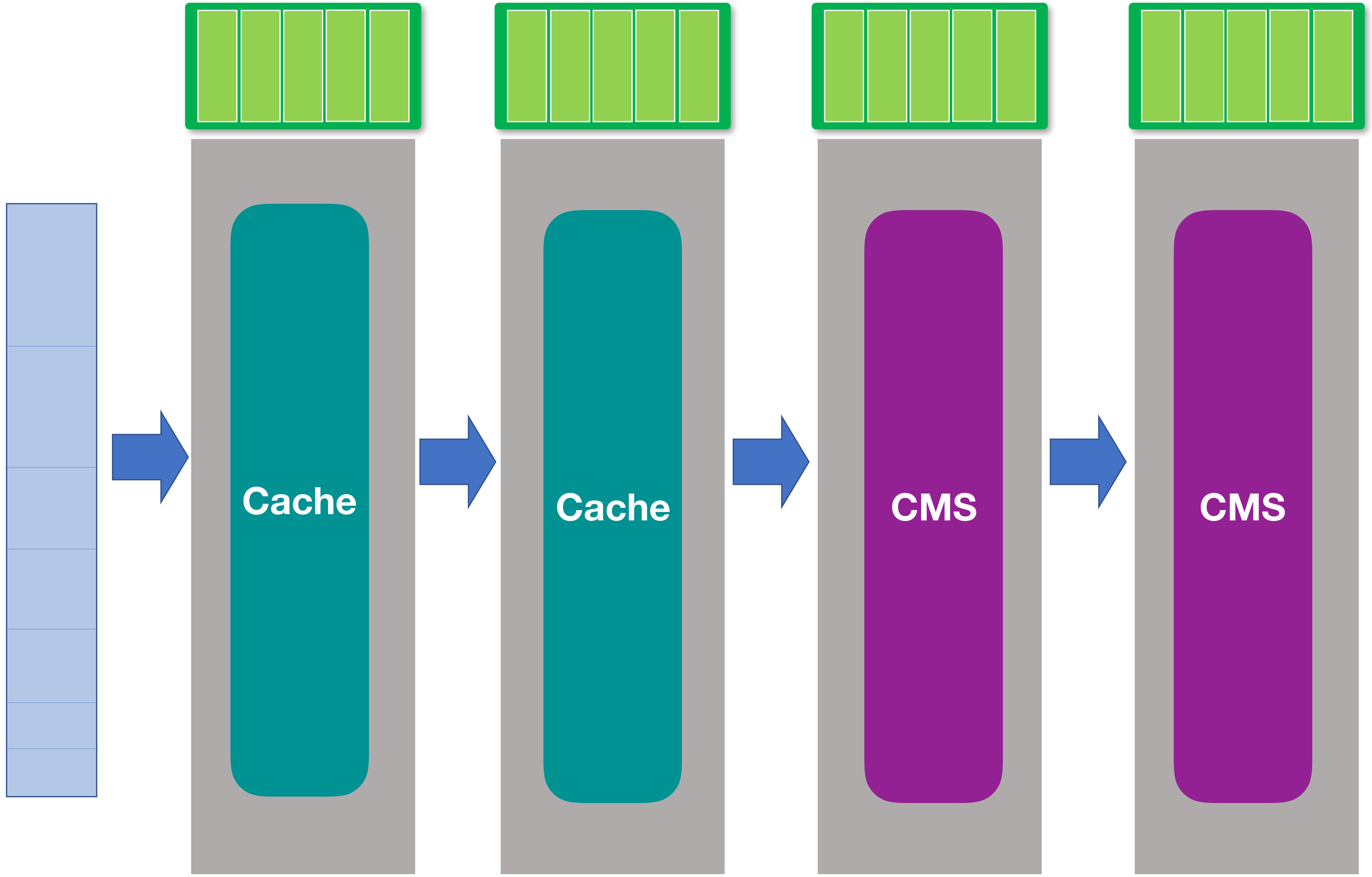
Cache of popular keys

If requests for key > 80, insert into cache

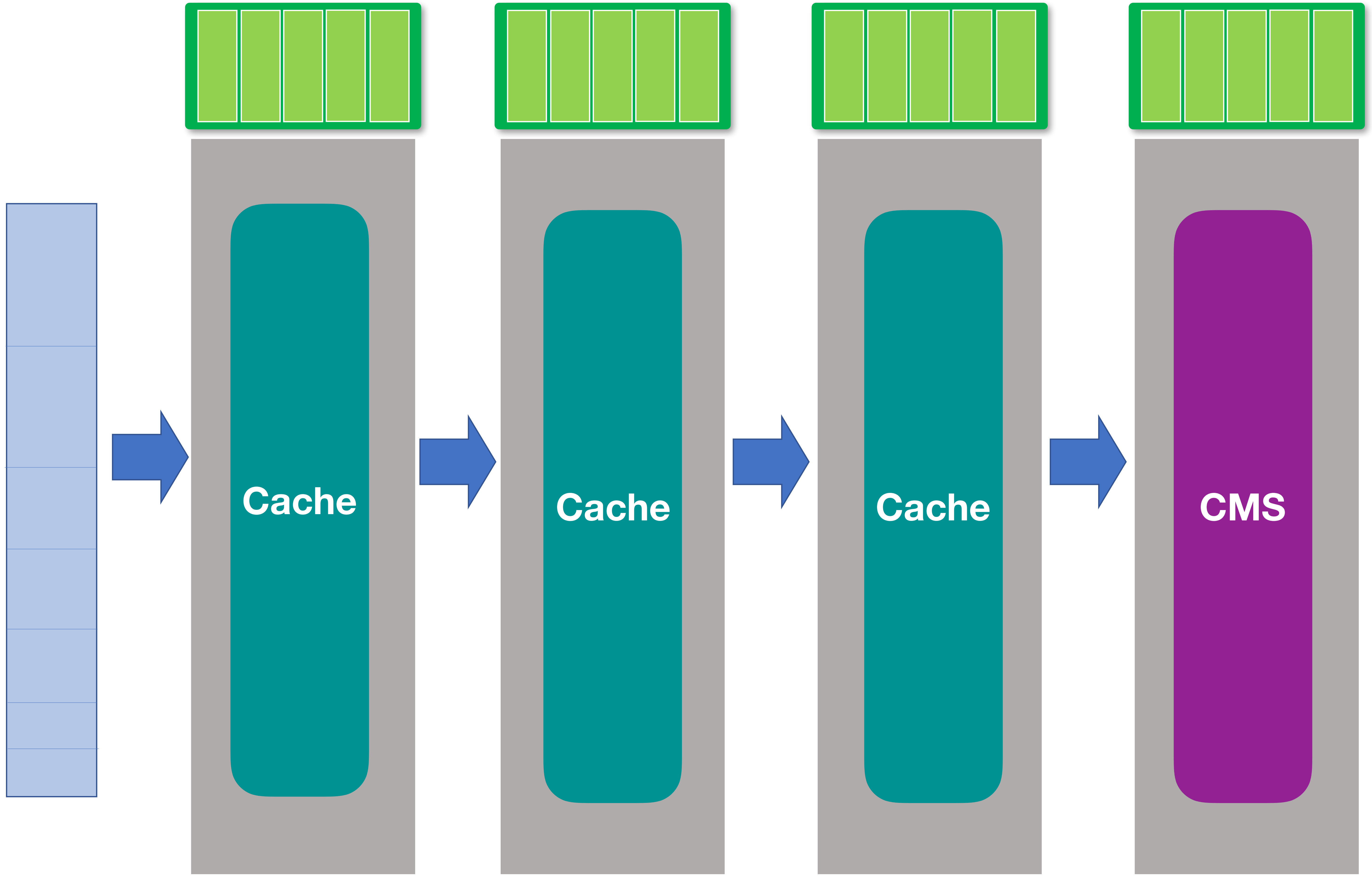


CMS

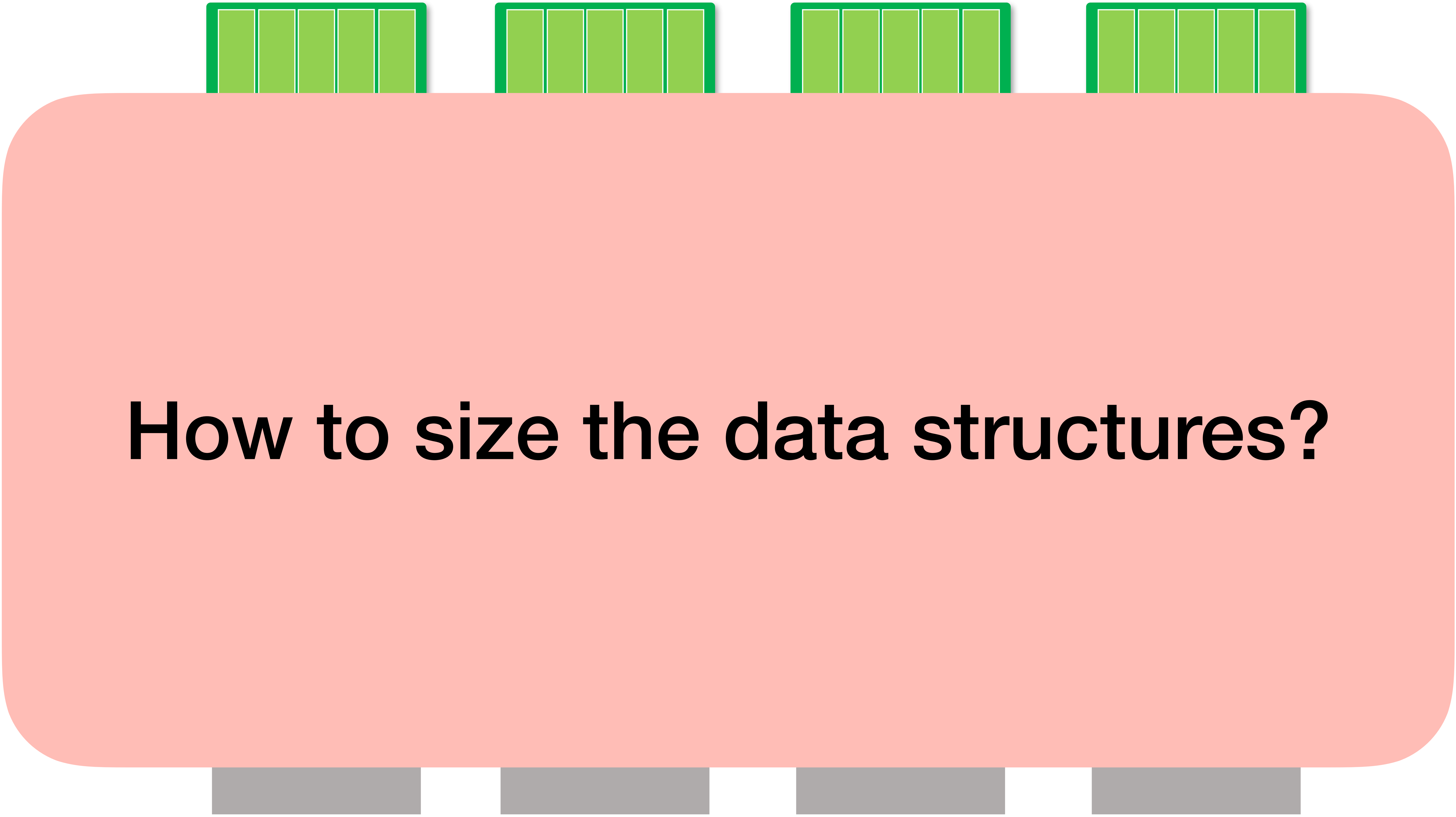
PISA



PISA

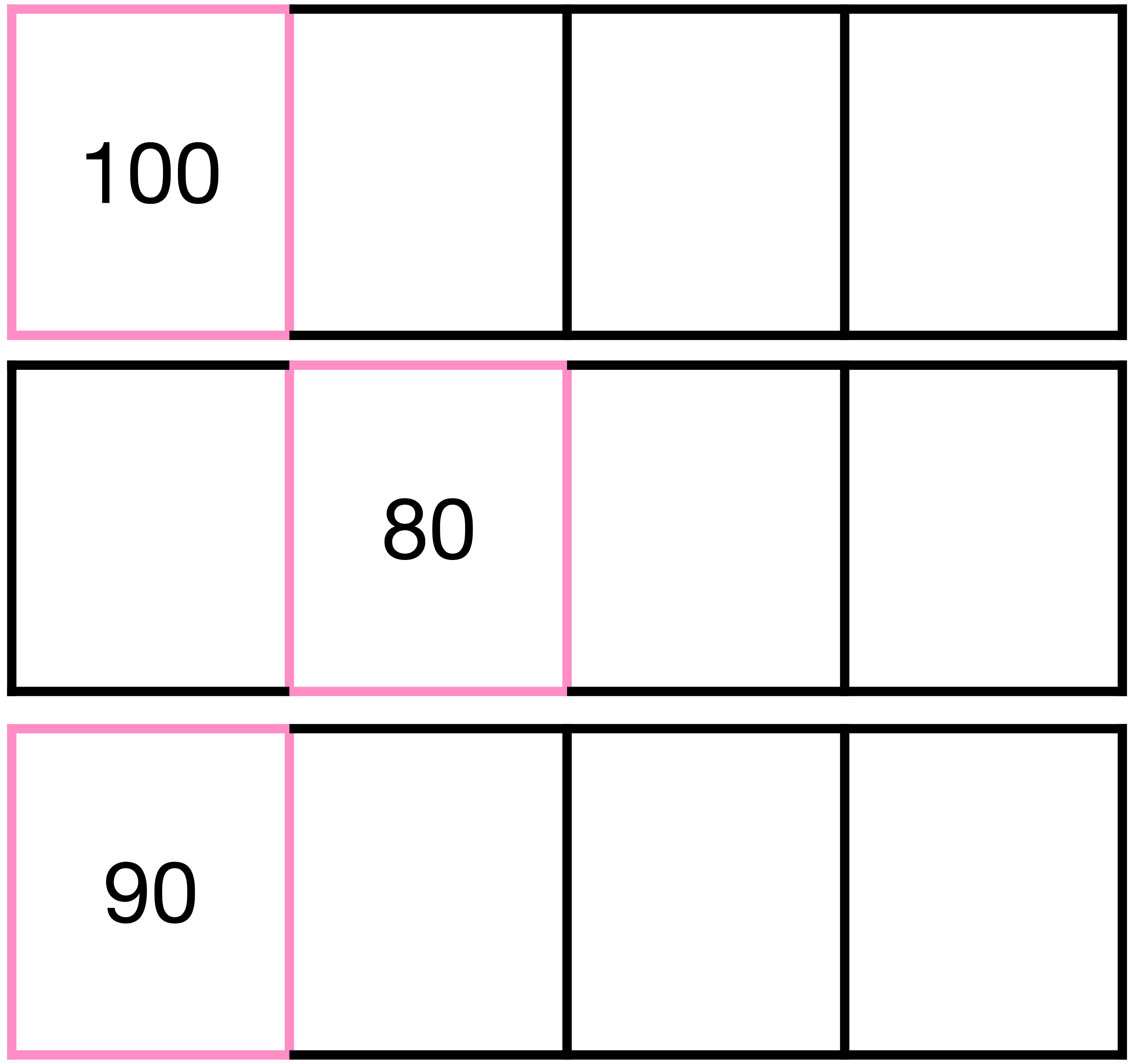


PISA



How to size the data structures?

Resources vs Accuracy

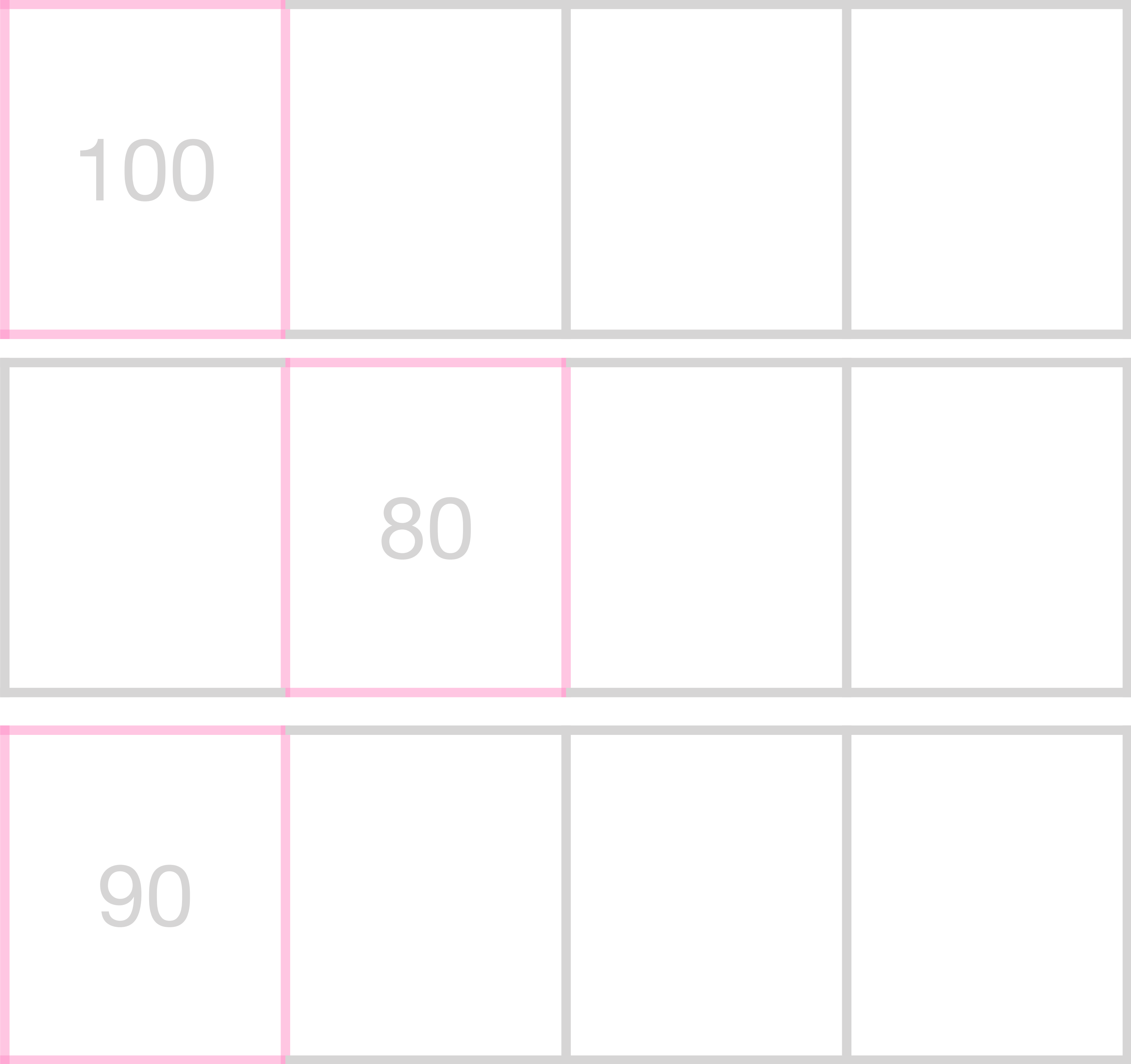


100			
	80		
90			

Actual count(x) = 50

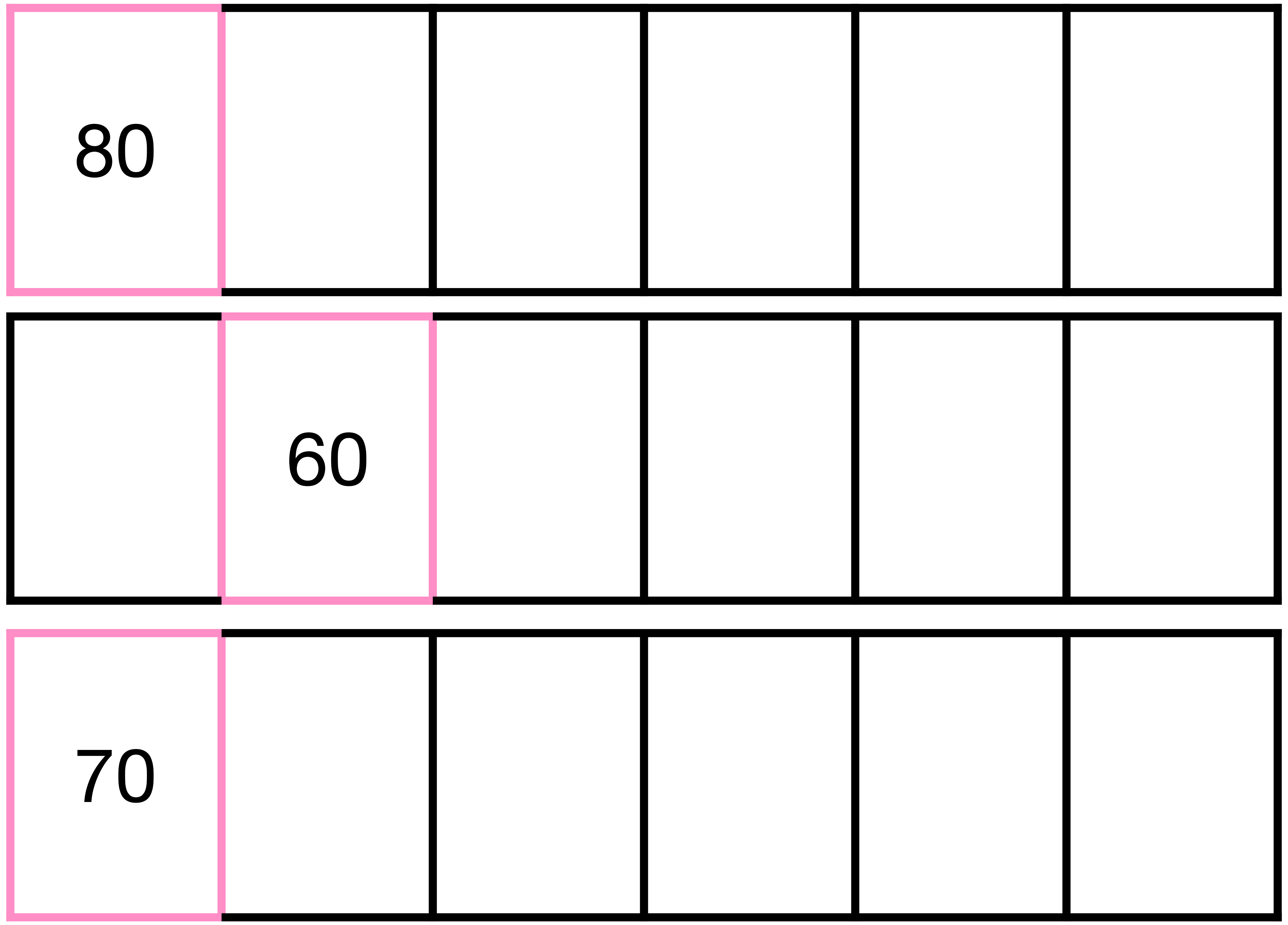
Estimated count(x) = 80

Resources vs Accuracy



Actual count(x) = 50

Estimated count(x) = 60



Outline

Elastic Structures

P4All

- Language
- Compiler
- Evaluation

Ongoing + Future Work

Outline

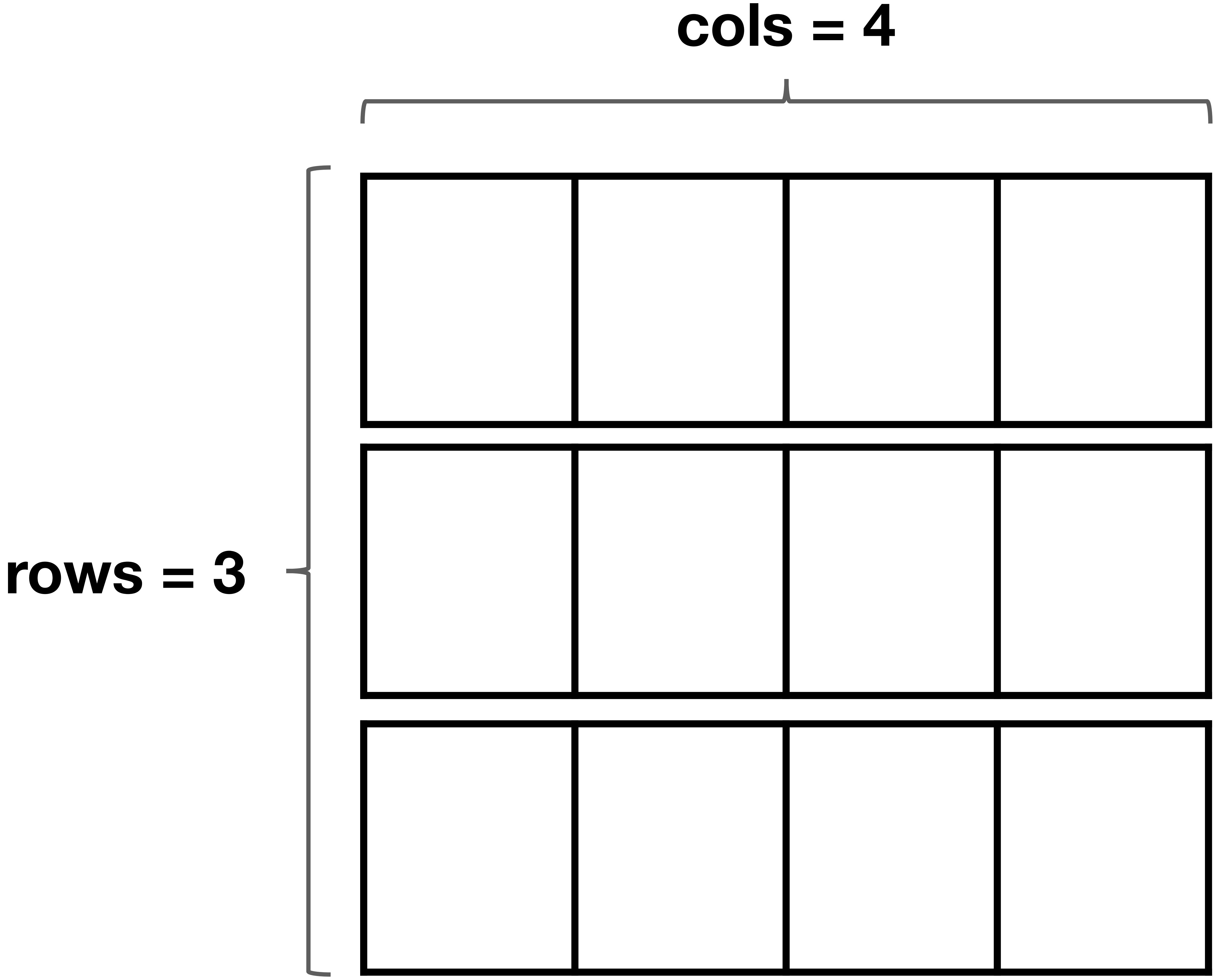
Elastic Structures

P4All

- Language
- Compiler
- Evaluation

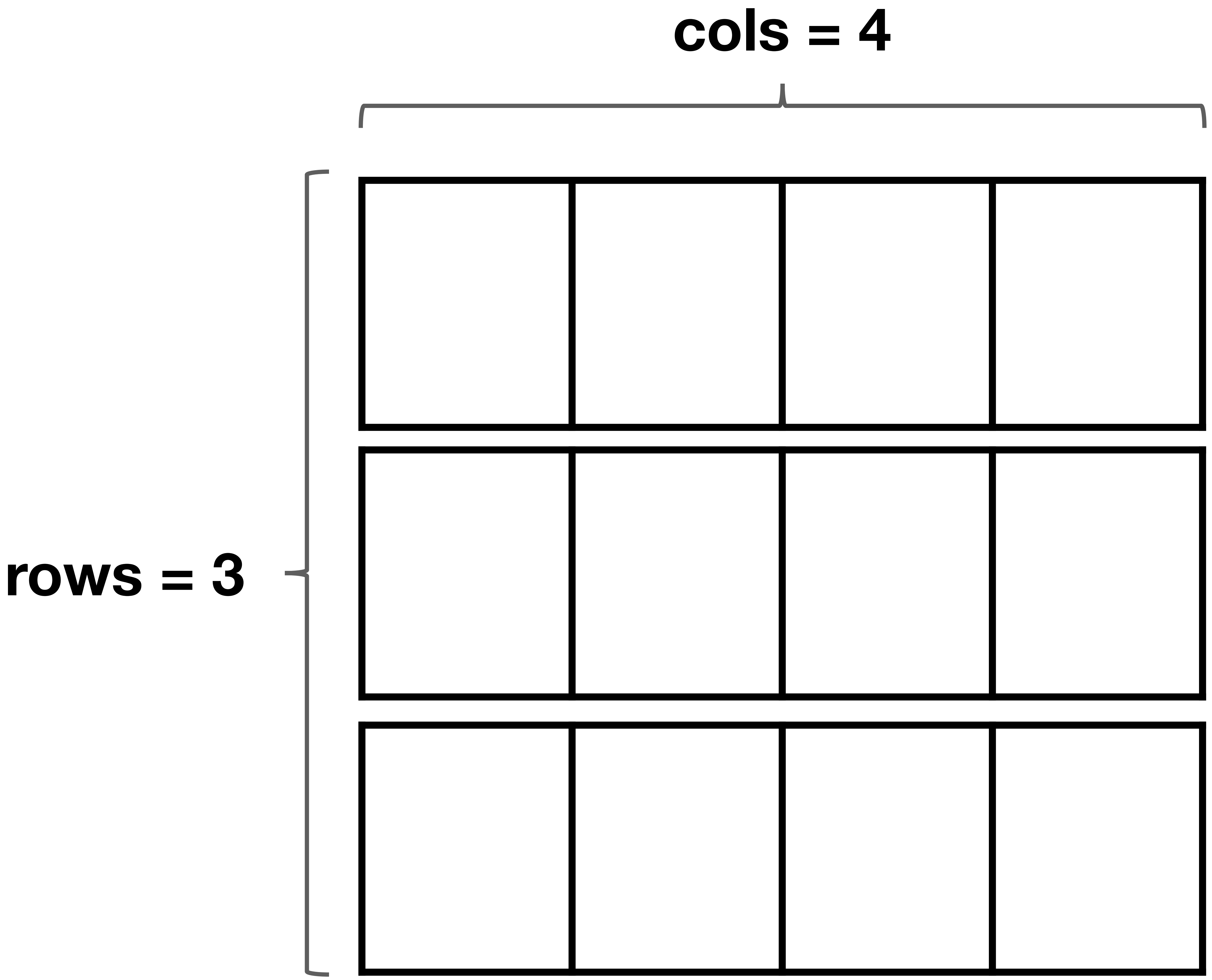
Ongoing + Future Work

Symbolic Values

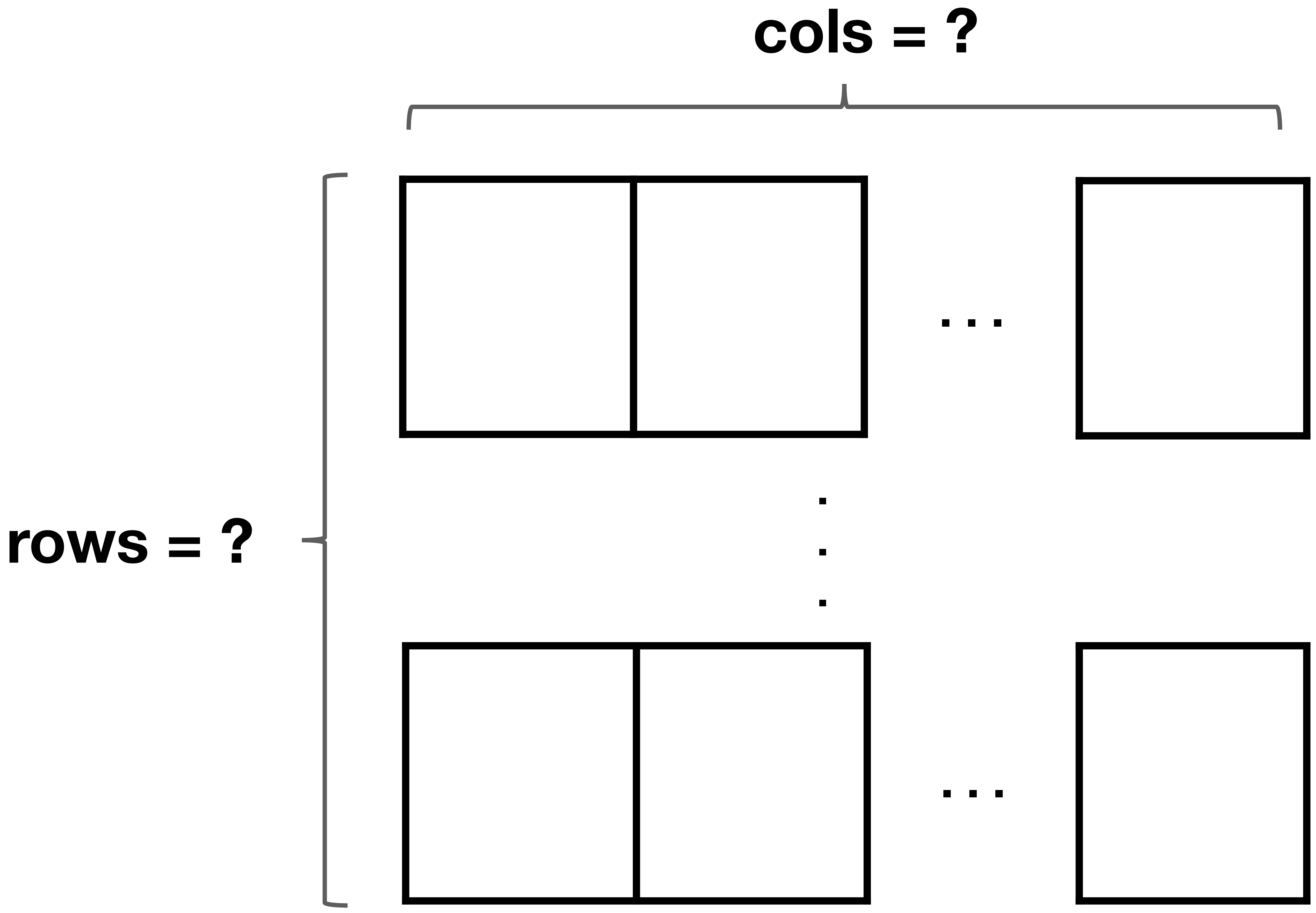


Symbolic Values

```
register<bit<32>>(4) row1;
register<bit<32>>(4) row2;
register<bit<32>>(4) row3;
```

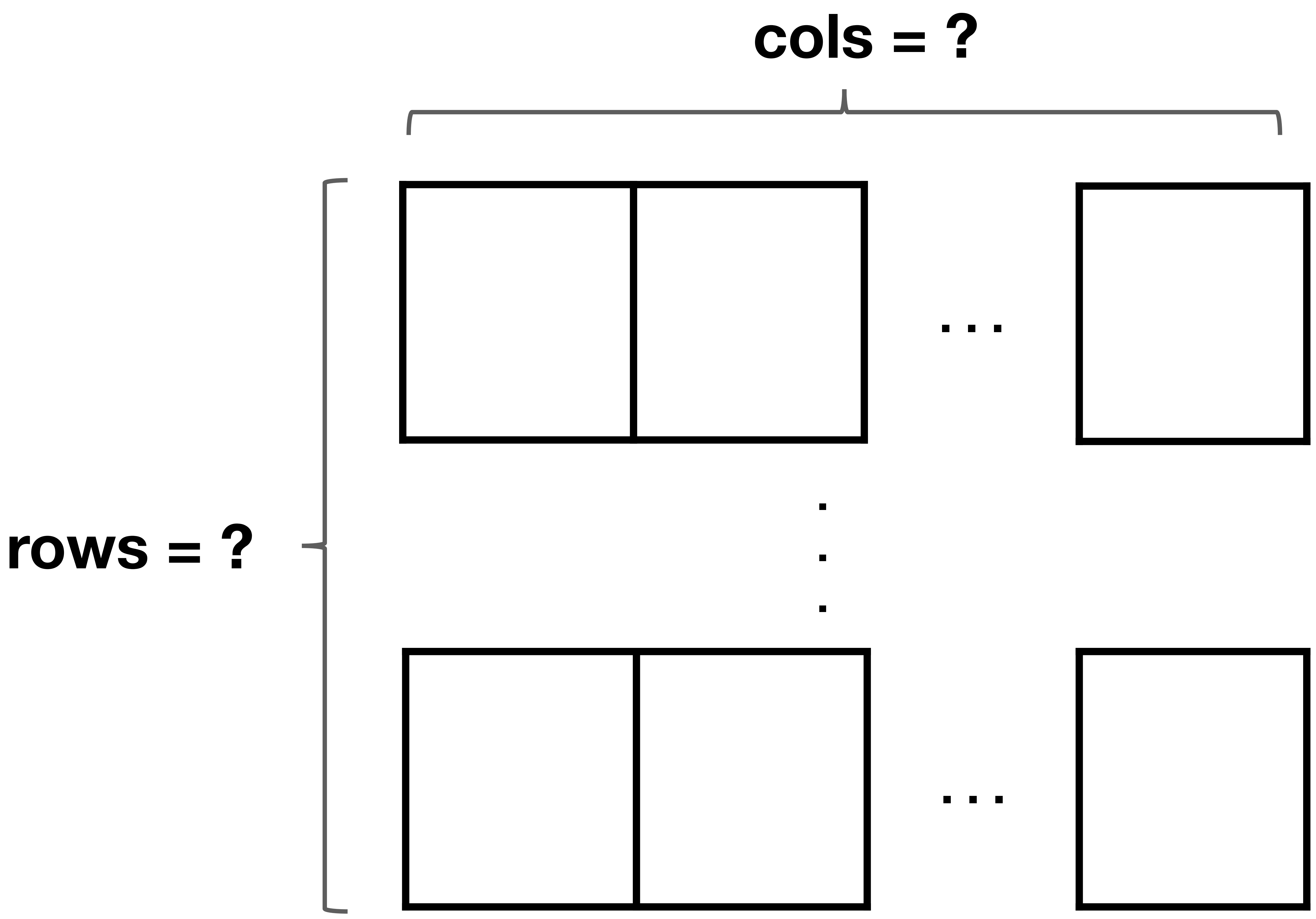


Symbolic Values

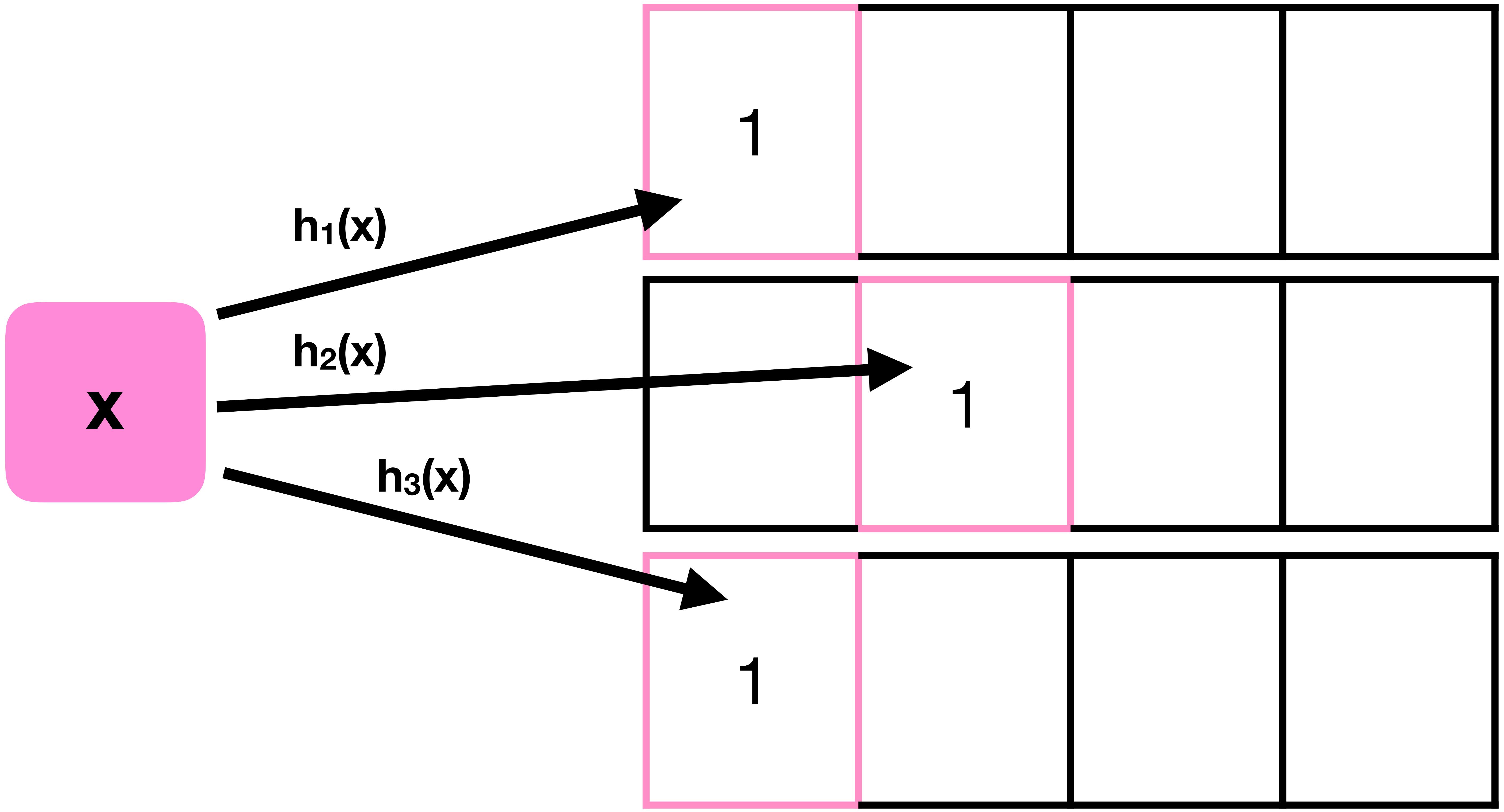


Symbolic Values

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

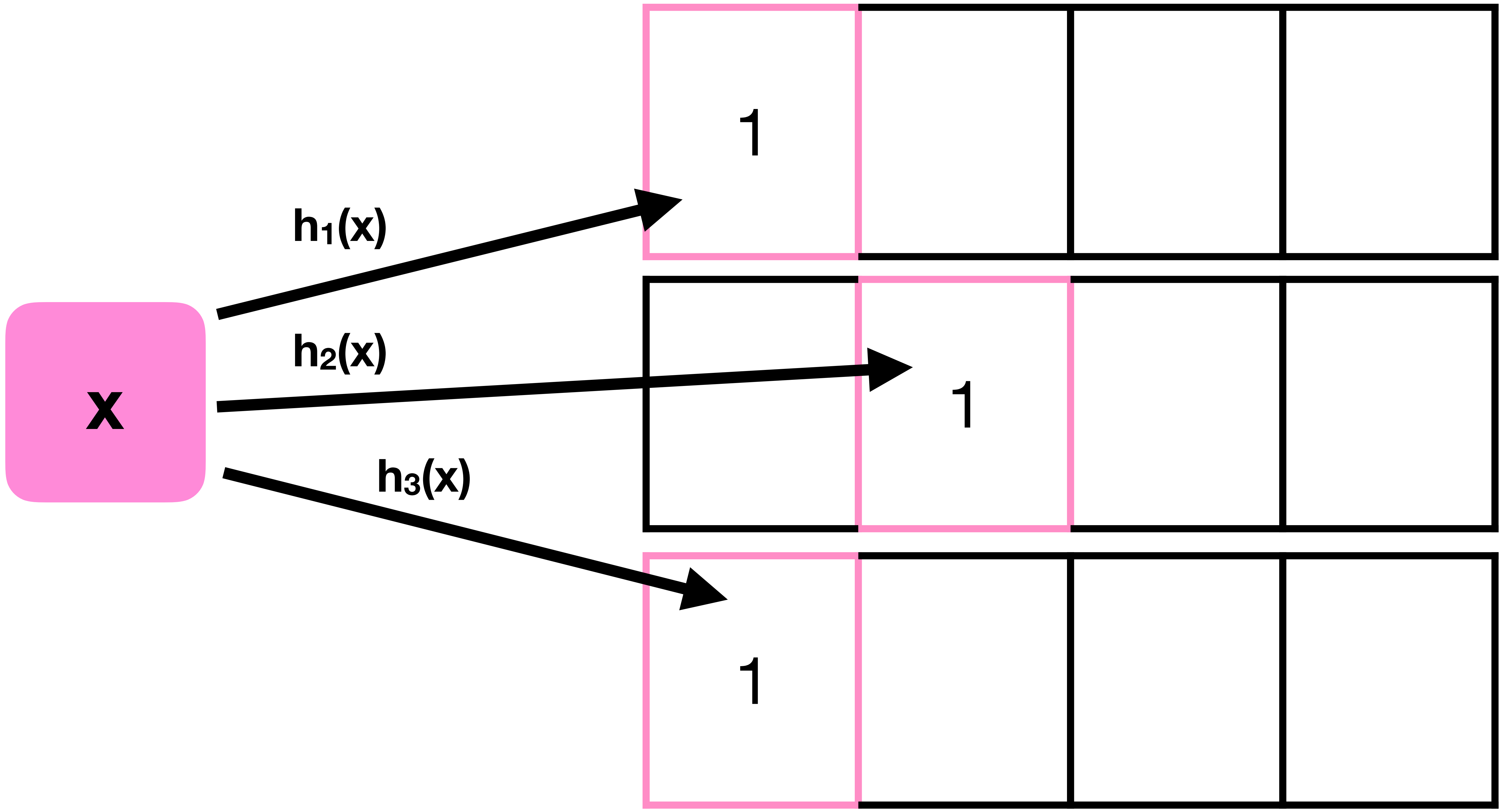


For Loops

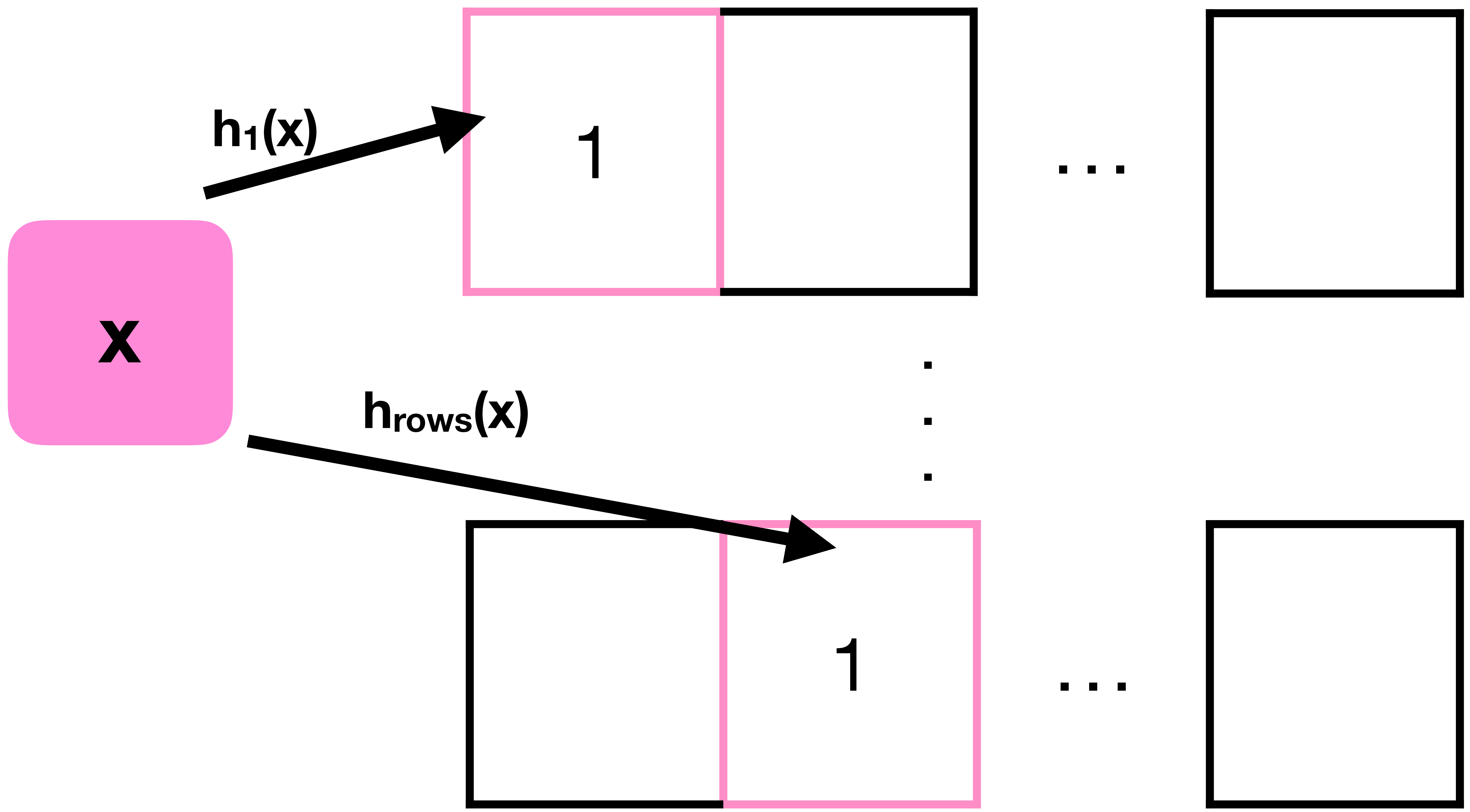


For Loops

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

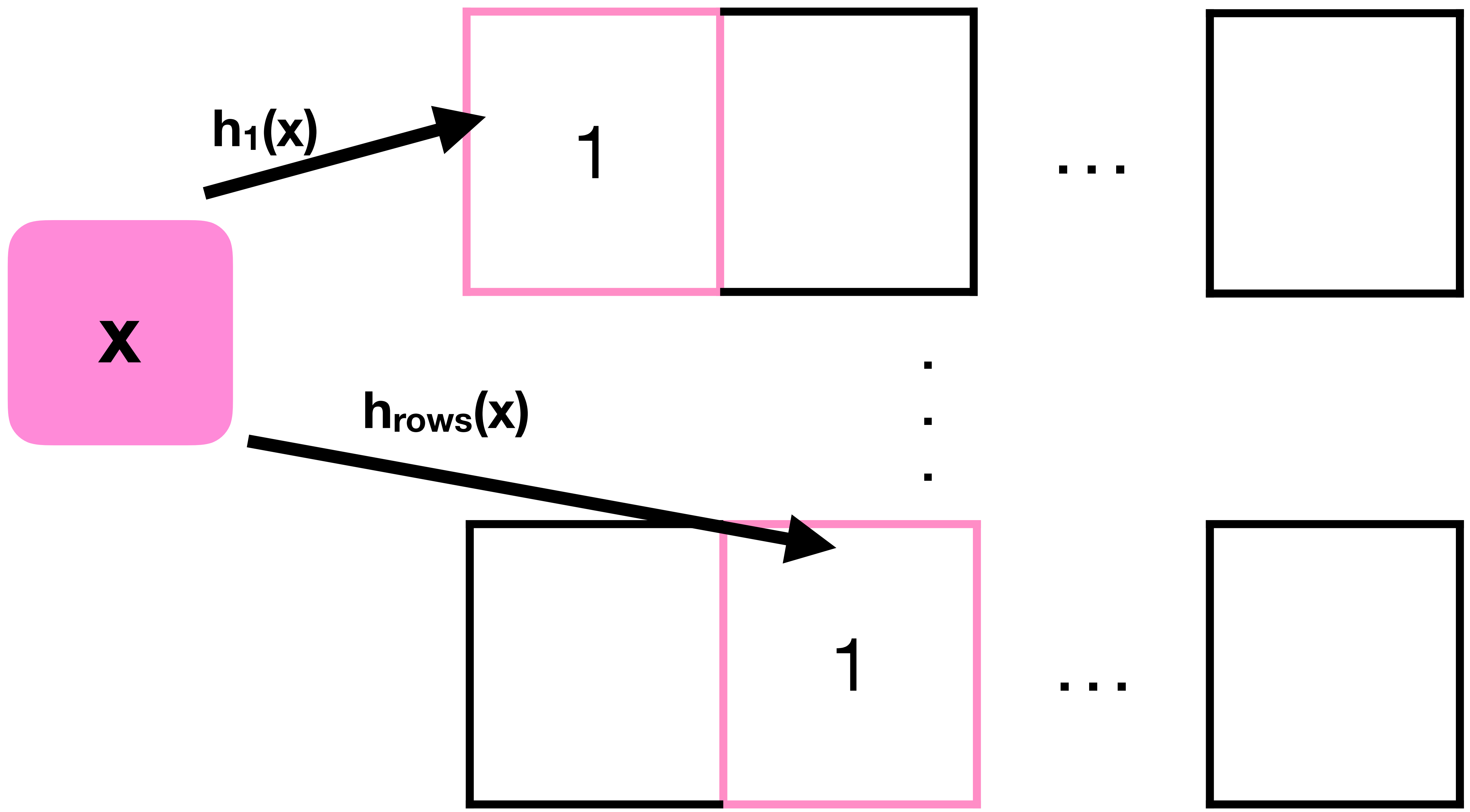


For Loops

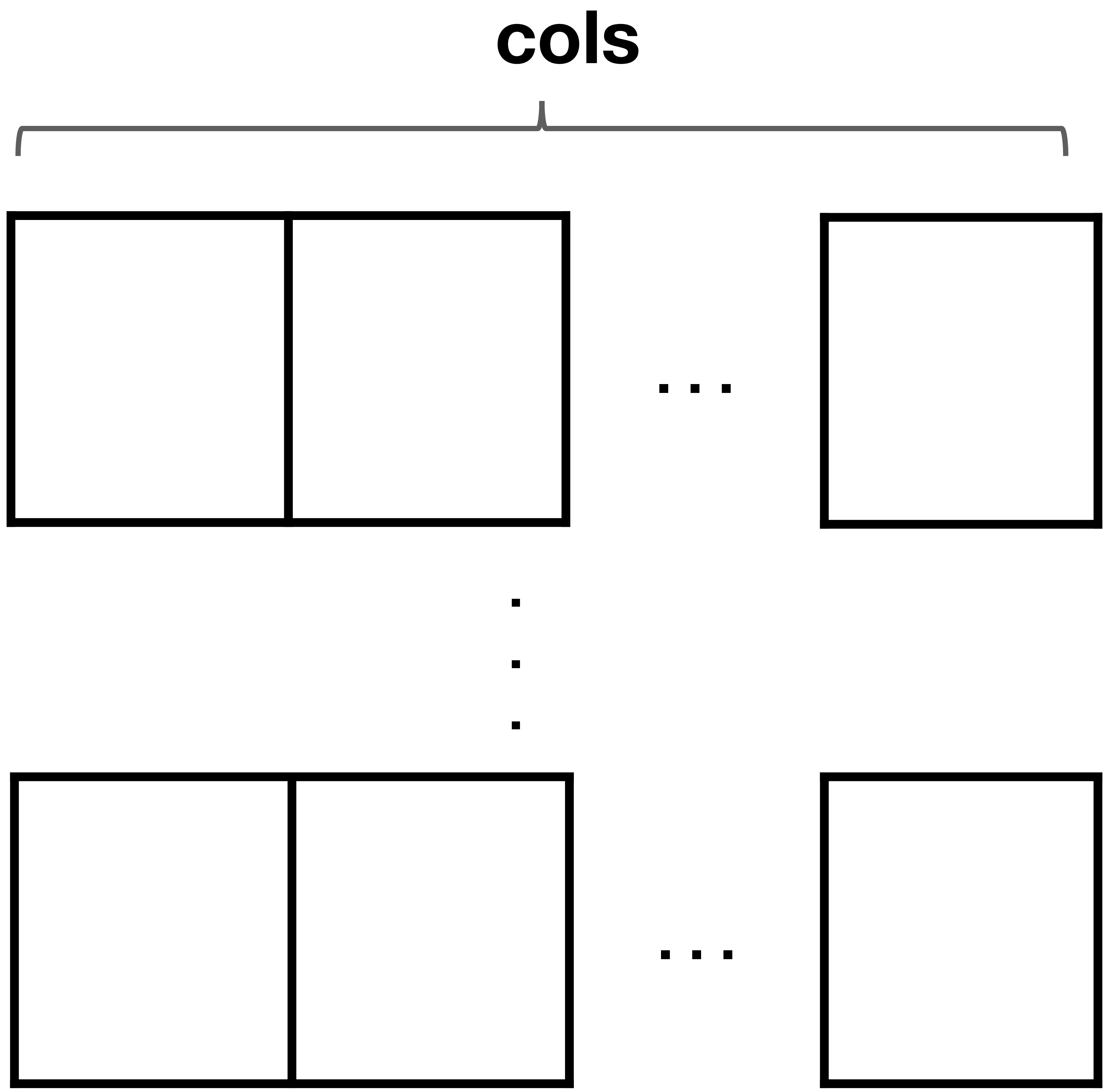


For Loops

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



Objective Functions

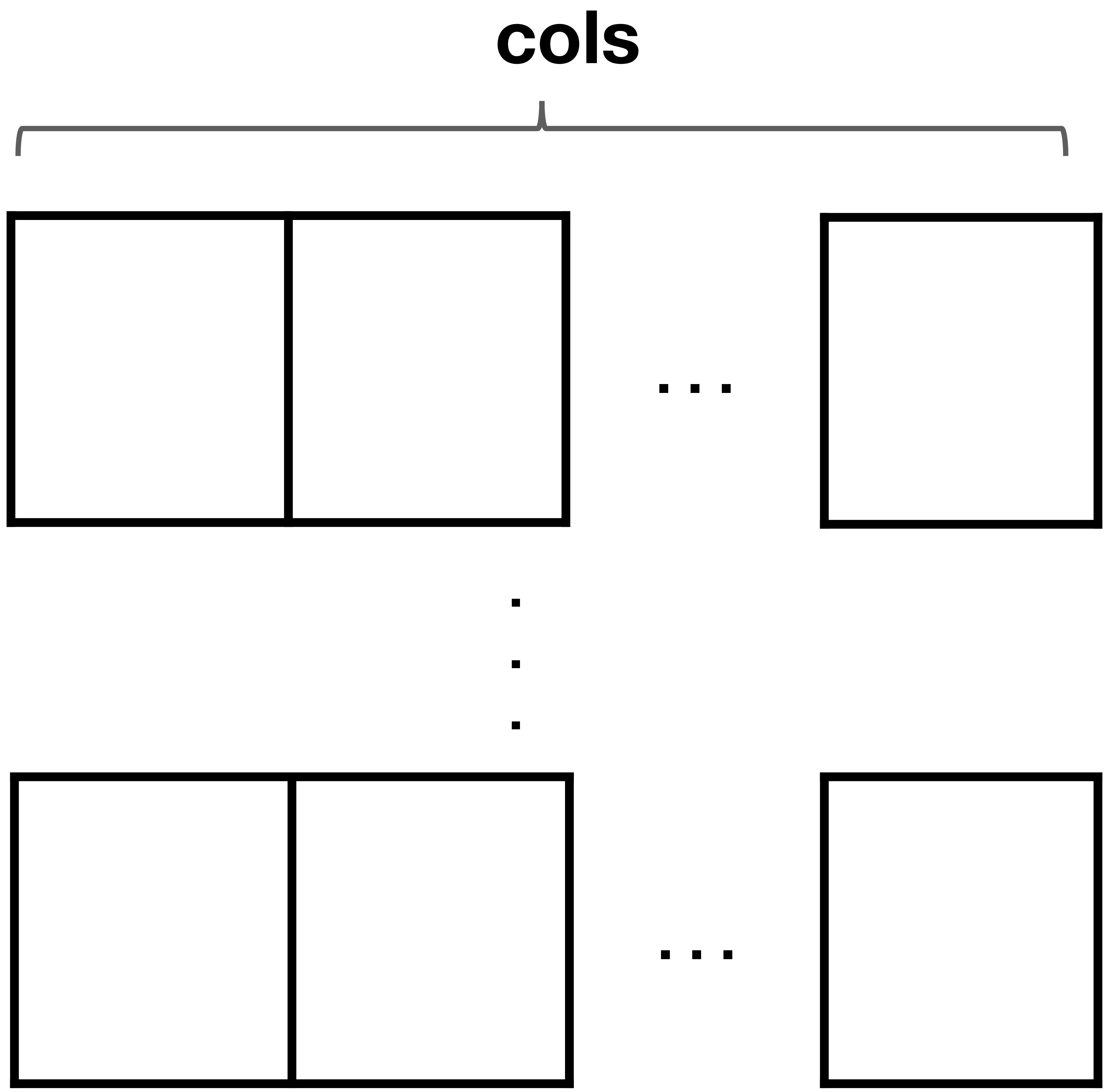


f(cols) = CMS error

Objective Functions

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

f(cols) = CMS error



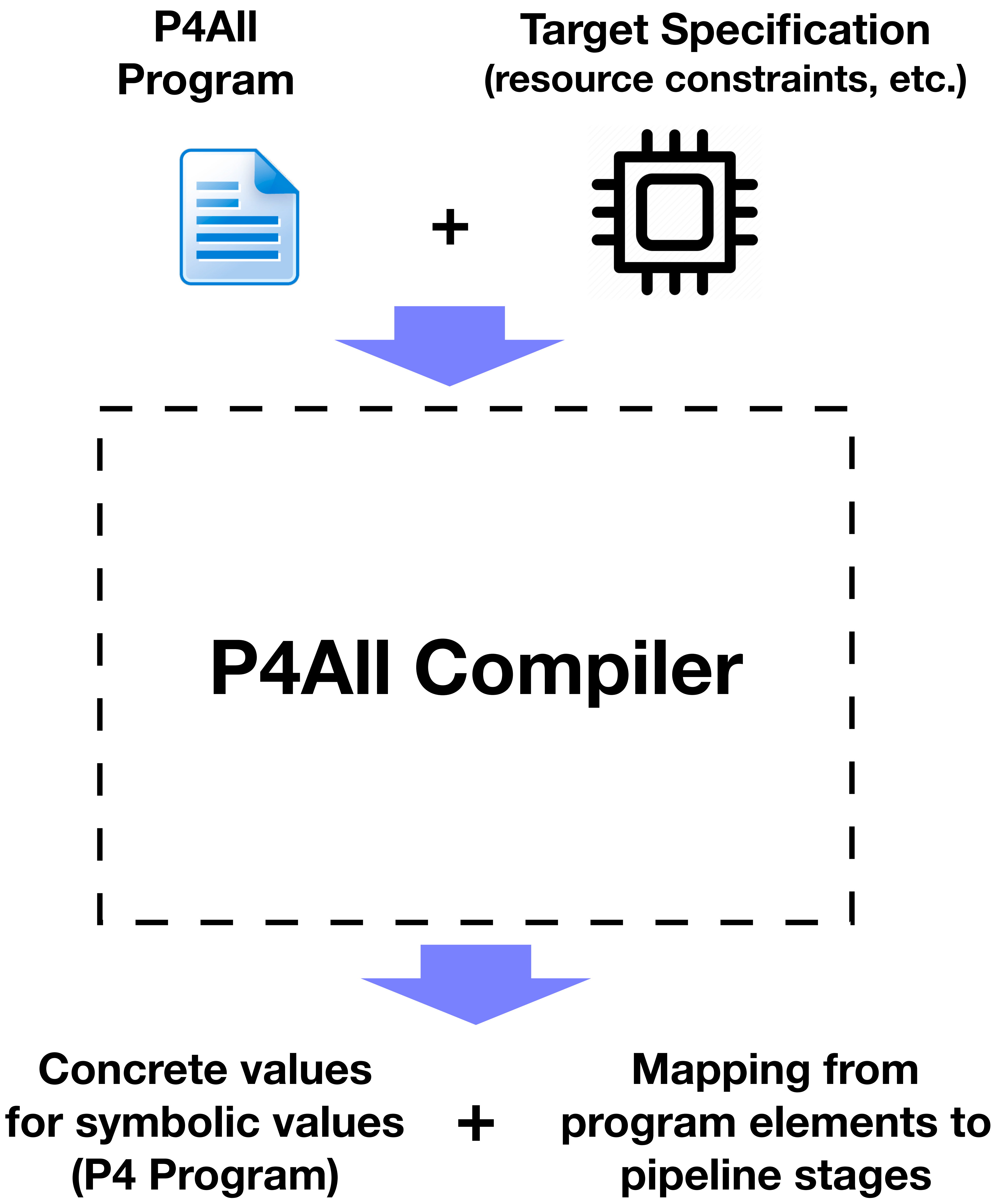
Outline

Elastic Structures

P4All

- Language
- Compiler
- Evaluation

Ongoing + Future Work



P4All Program

+

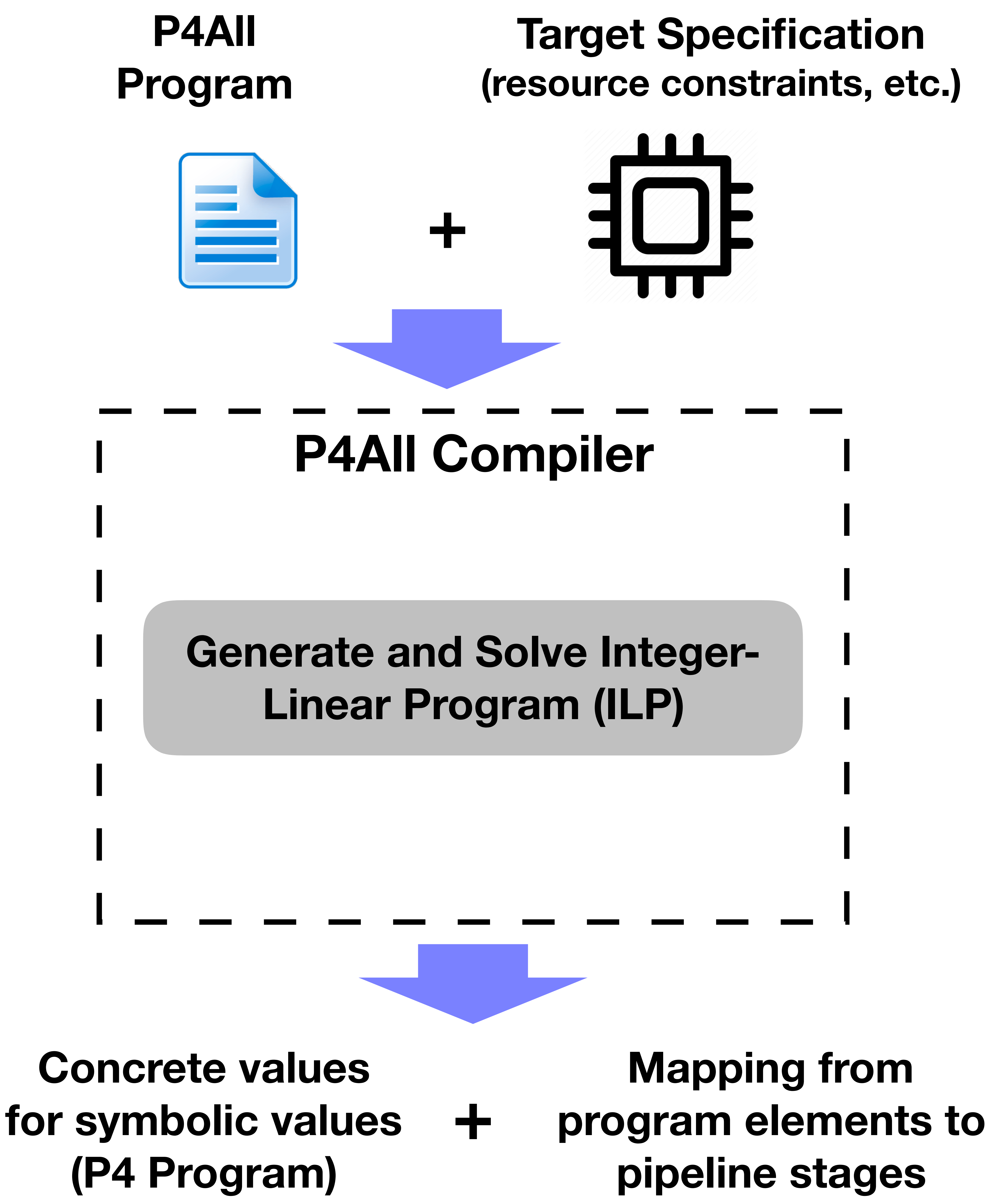
Target Specification (resource constraints, etc.)

P4All Compiler

Concrete values for symbolic values (P4 Program)

+

Mapping from program elements to pipeline stages



P4All Program

+

Target Specification (resource constraints, etc.)

P4All Compiler

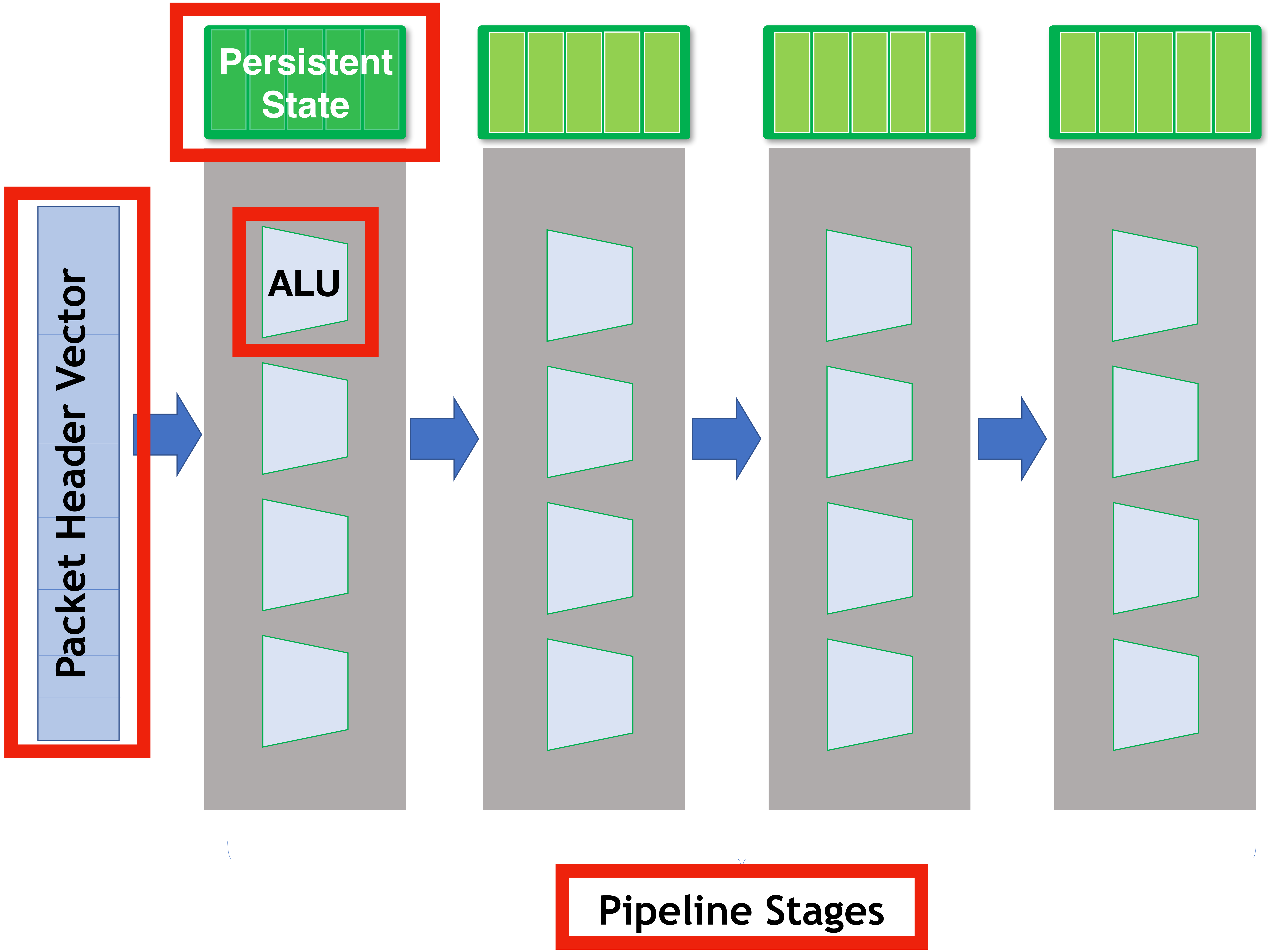
Generate and Solve Integer-Linear Program (ILP)

Concrete values for symbolic values (P4 Program)

+

Mapping from program elements to pipeline stages

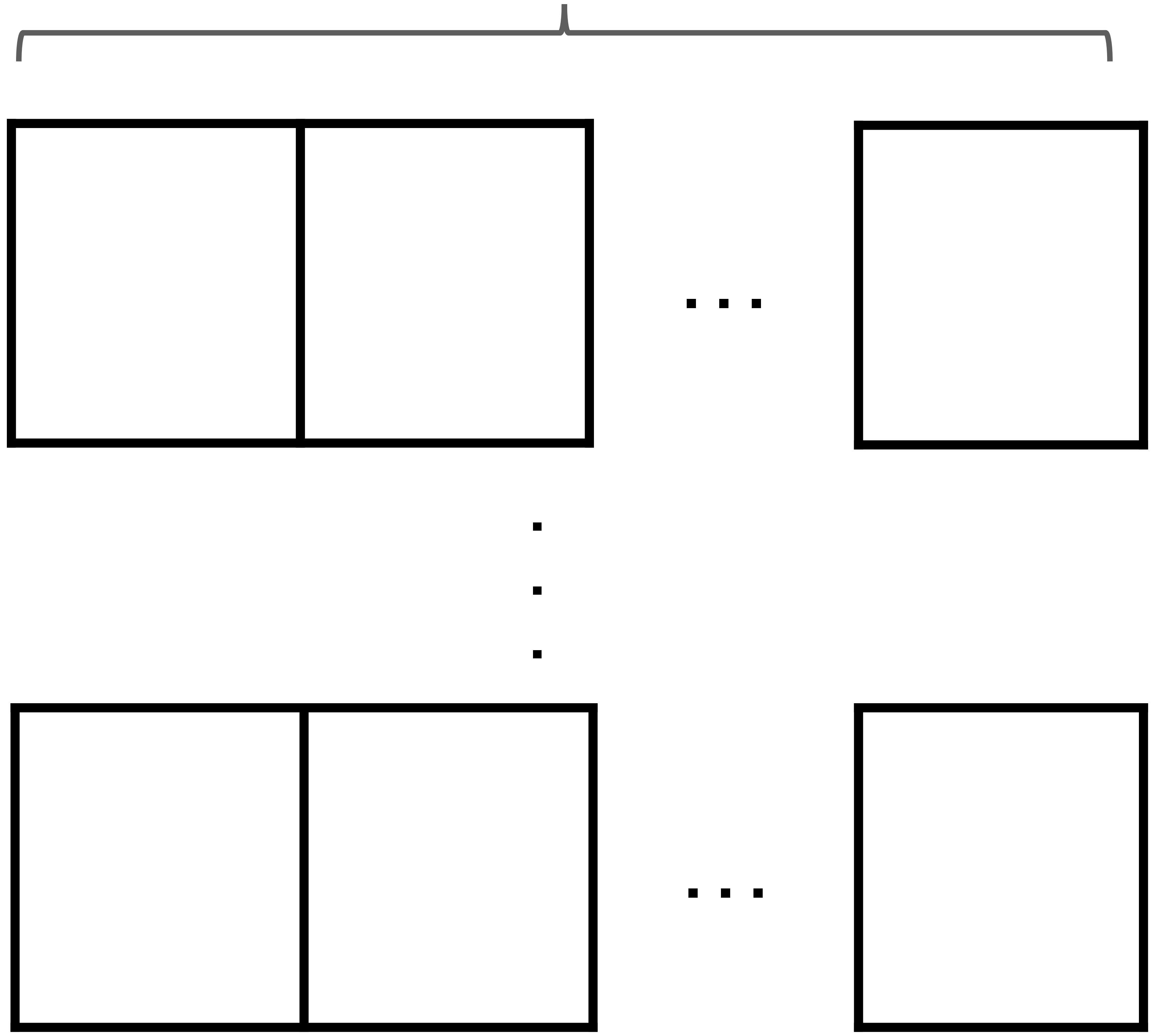
ILP Constraints



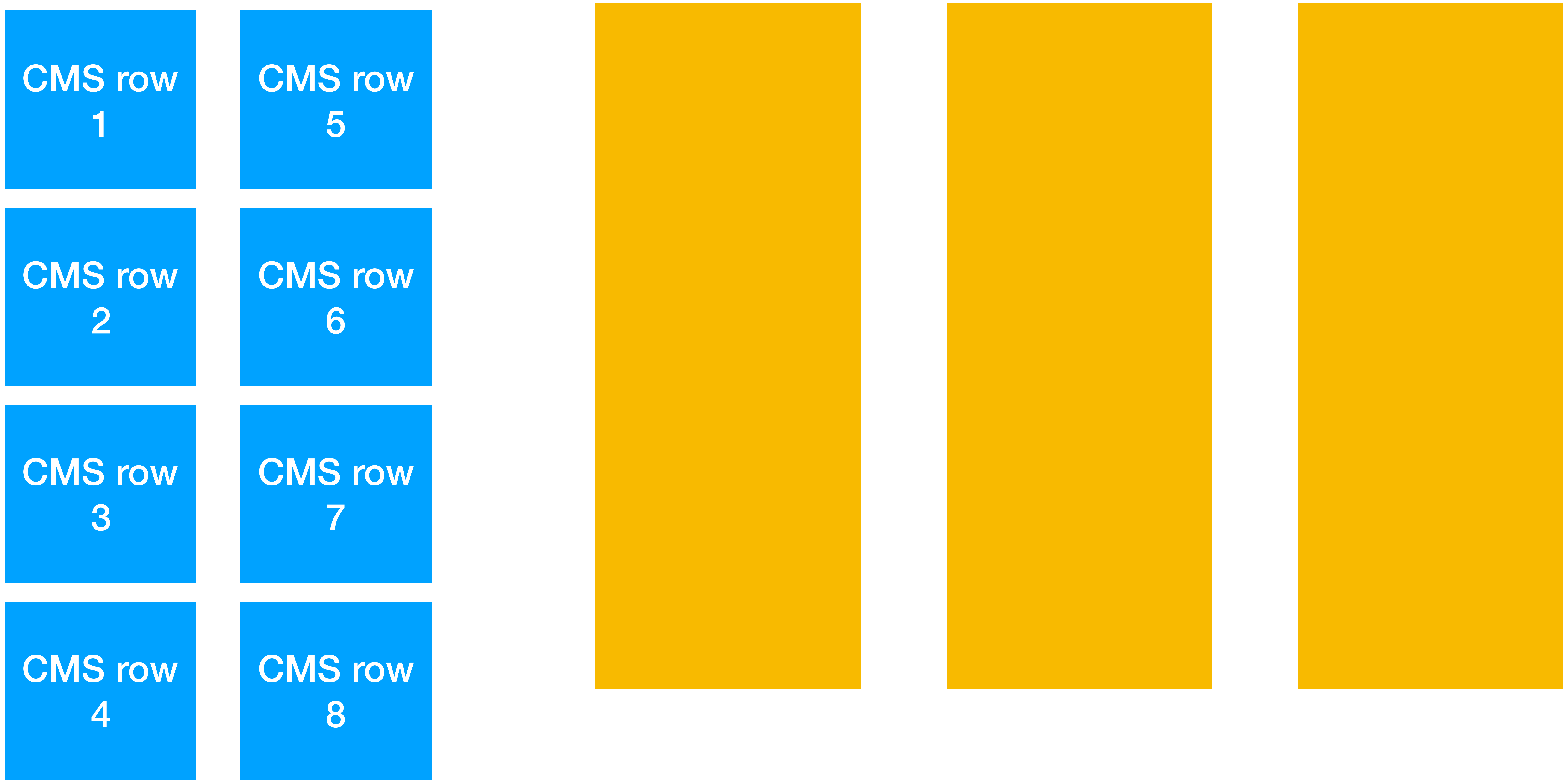
ILP Objective

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

f(cols) = CMS error



P4All Compiler



CMS row 1

CMS row 2

CMS row 3

CMS row 4

CMS row 5

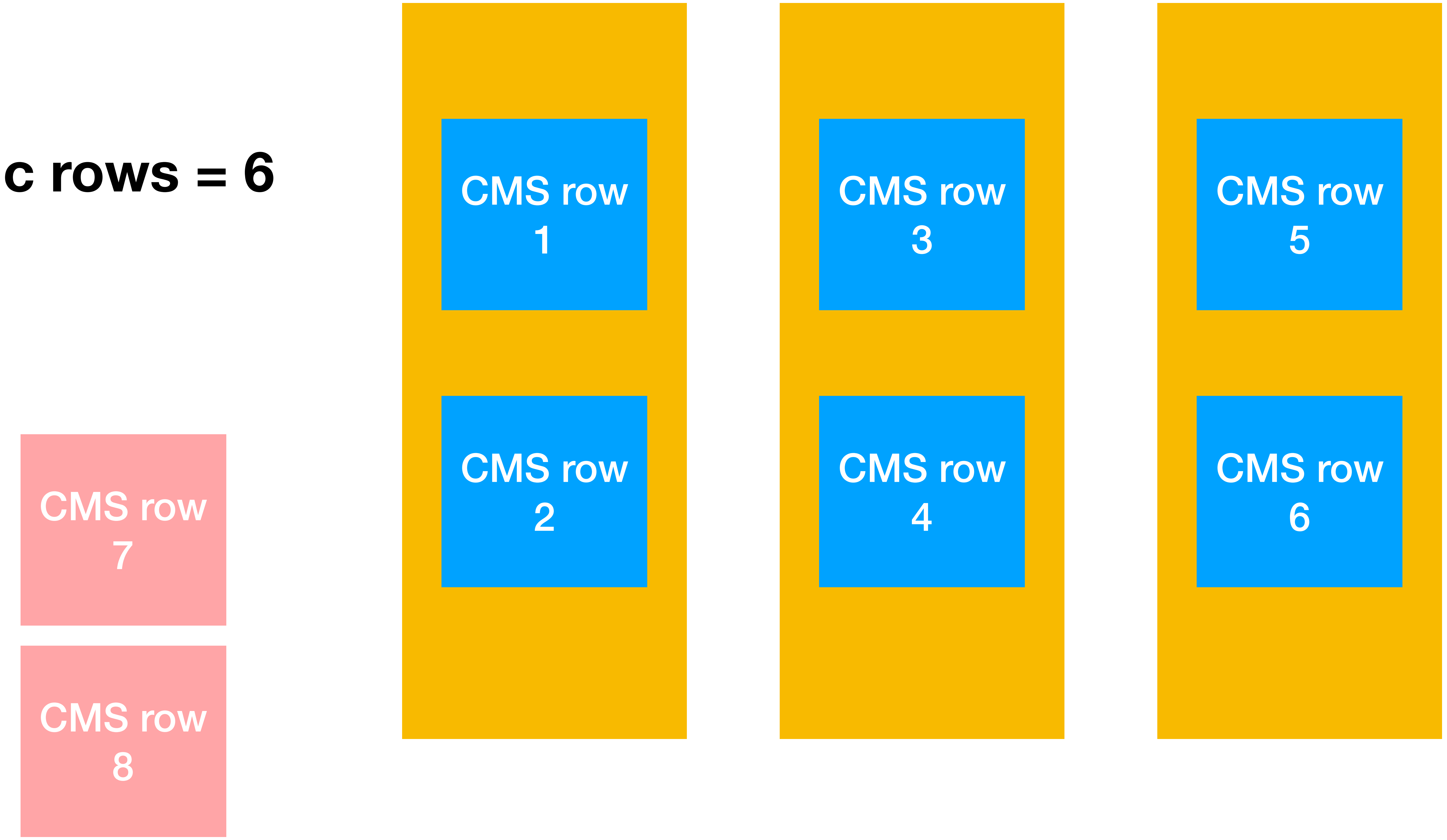
CMS row 6

CMS row 7

CMS row 8

P4All Compiler

symbolic rows = 6



Outline

Elastic Structures

P4All

Language

Compiler

Evaluation

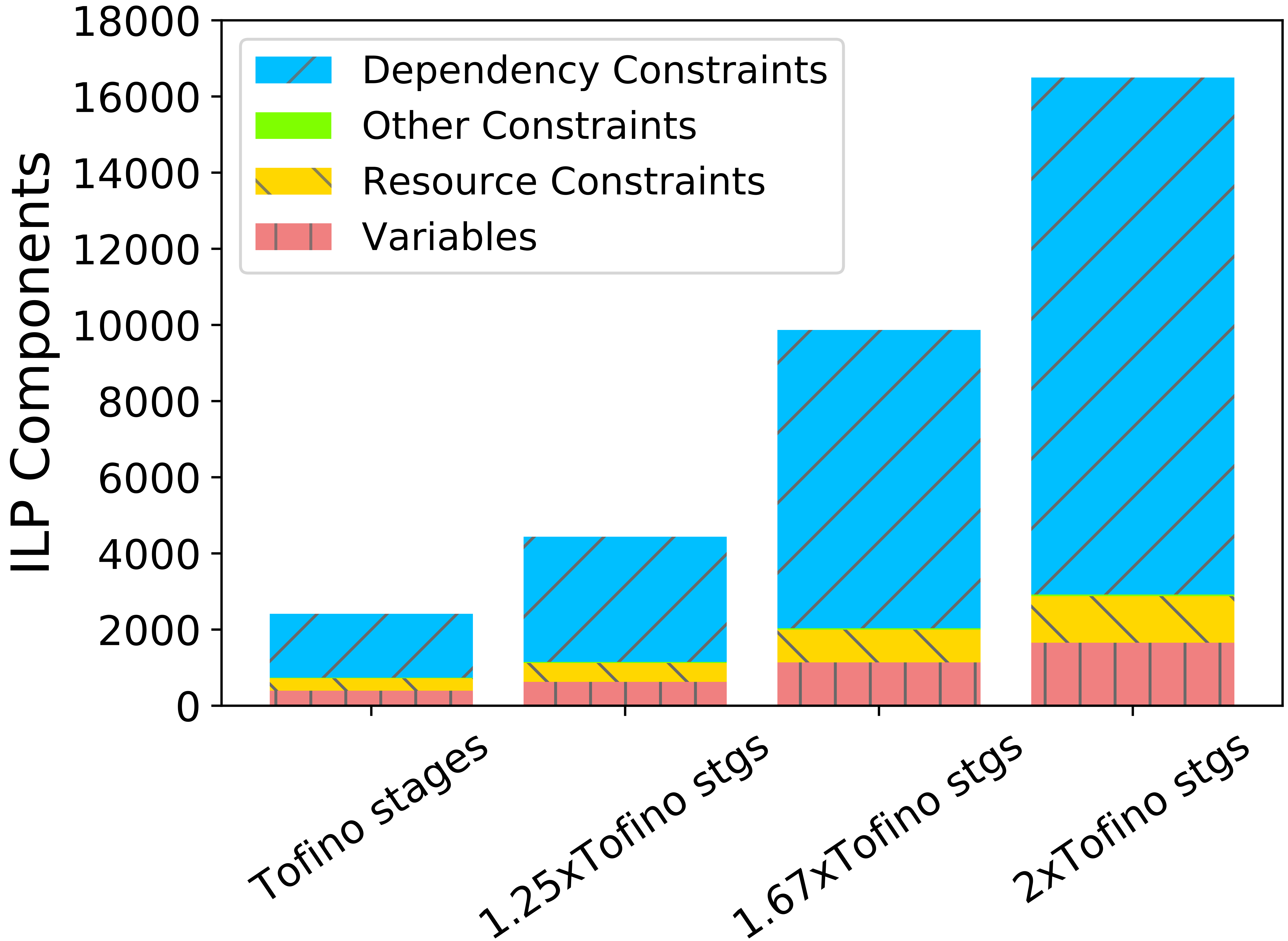
Ongoing + Future Work

P4All Applications

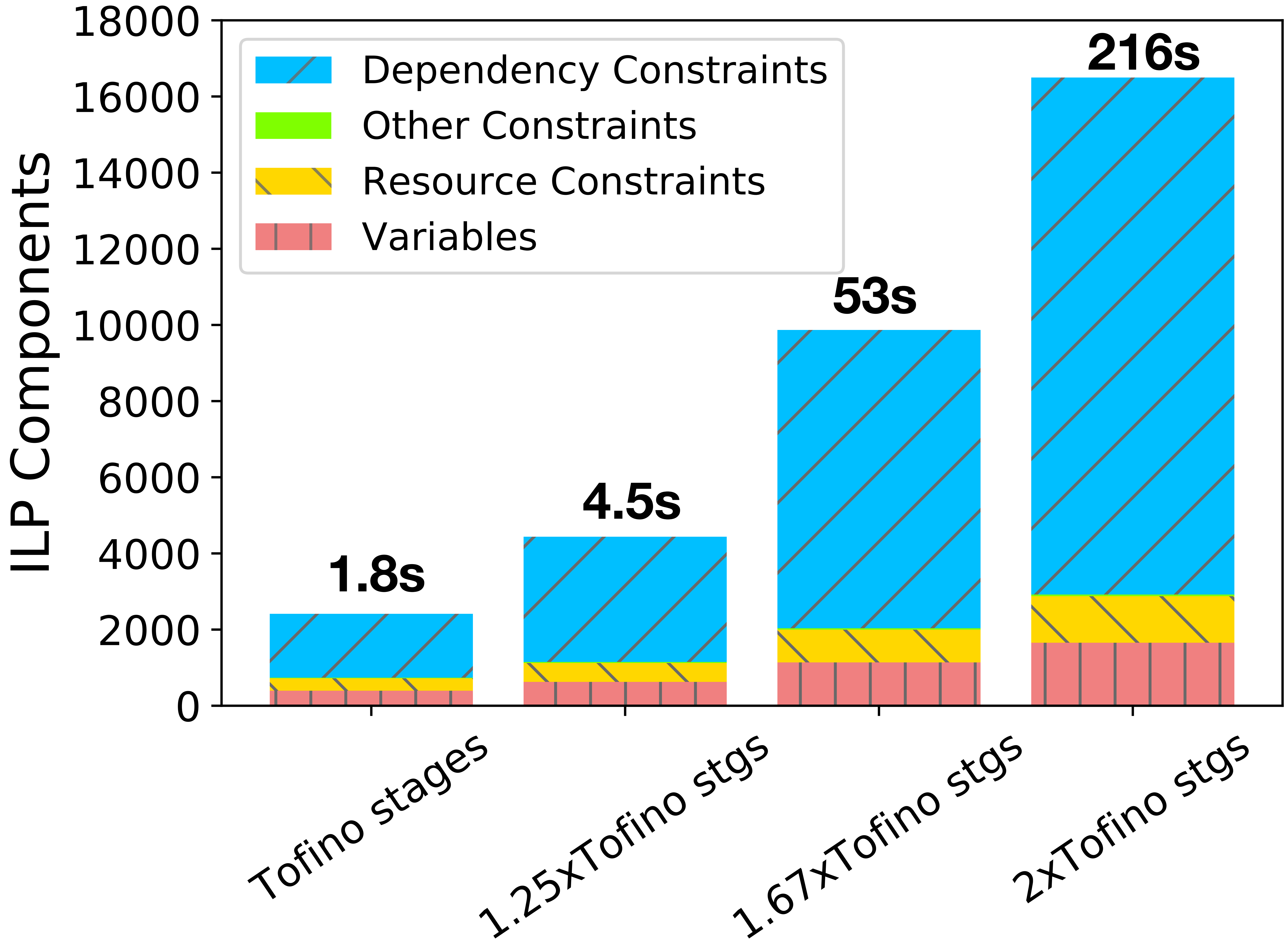
Application	Compile Time (s)
CMS	1.8
Key-value store	15.4
Key-value store + CMS	27.9
Switch.p4	0.2
IP forwarding + stateful firewall	0.4
Beaucoup	0.1
Precision	25.7
NetChain	27.9
SketchLearn	2.4
Conquest	5.8

ILP Performance

ILP Performance



ILP Performance



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