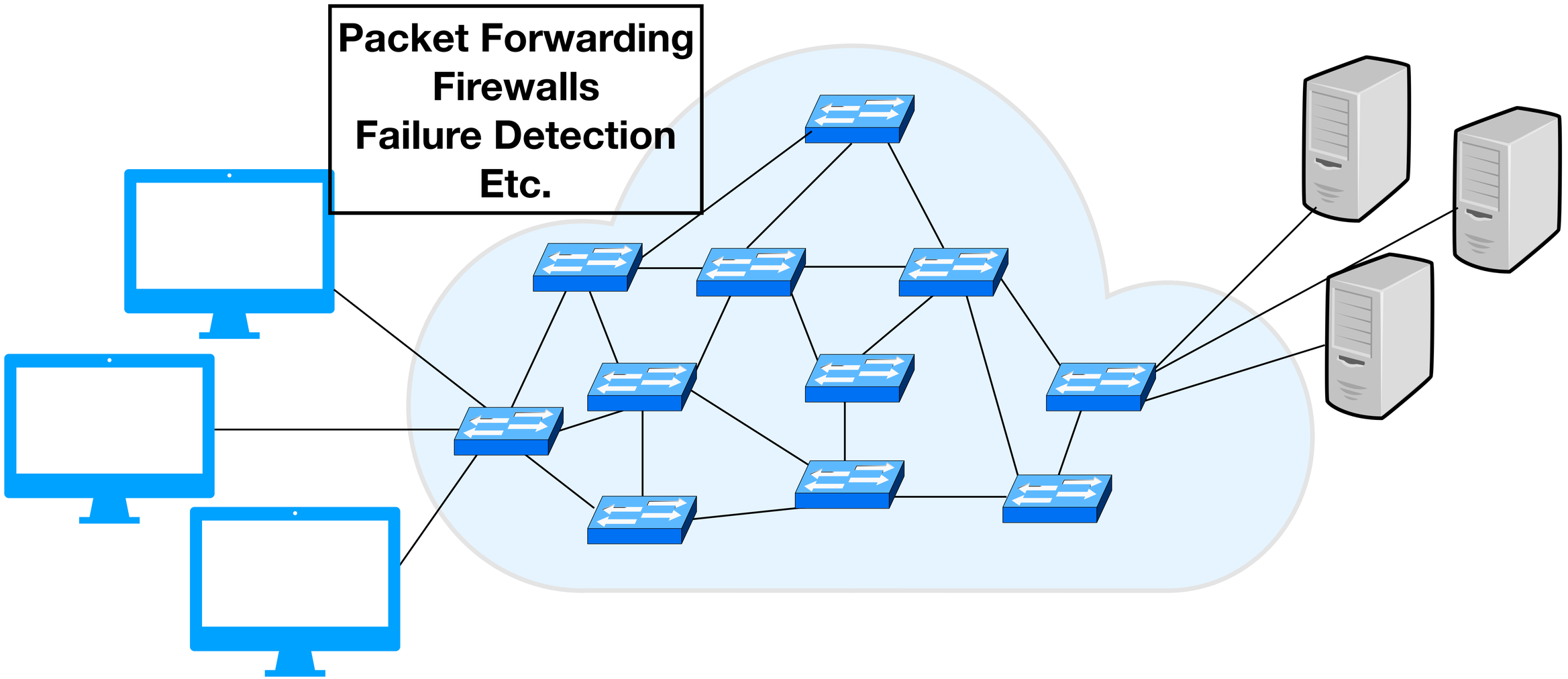
Language Expressiveness Under Extreme Scarcity in Programmable Data Planes

Mary Hogan

Switches must process ***billions*** of packets per second



We need specialized hardware to process at high speeds

Programmable switches

✓ **Support network programs (firewalls, packet forwarding, etc.)**

✓ **Fast processing**

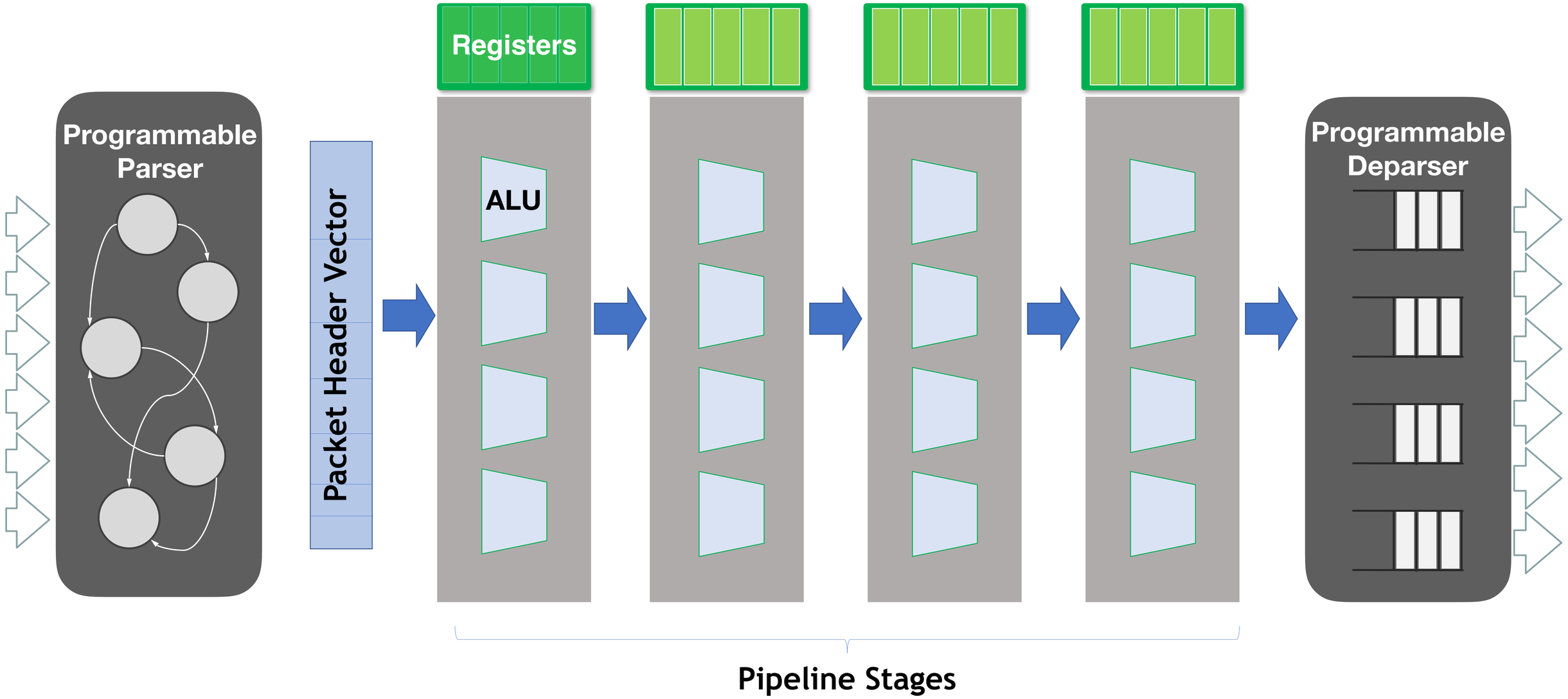
No free lunch…

(1) Processing restrictions

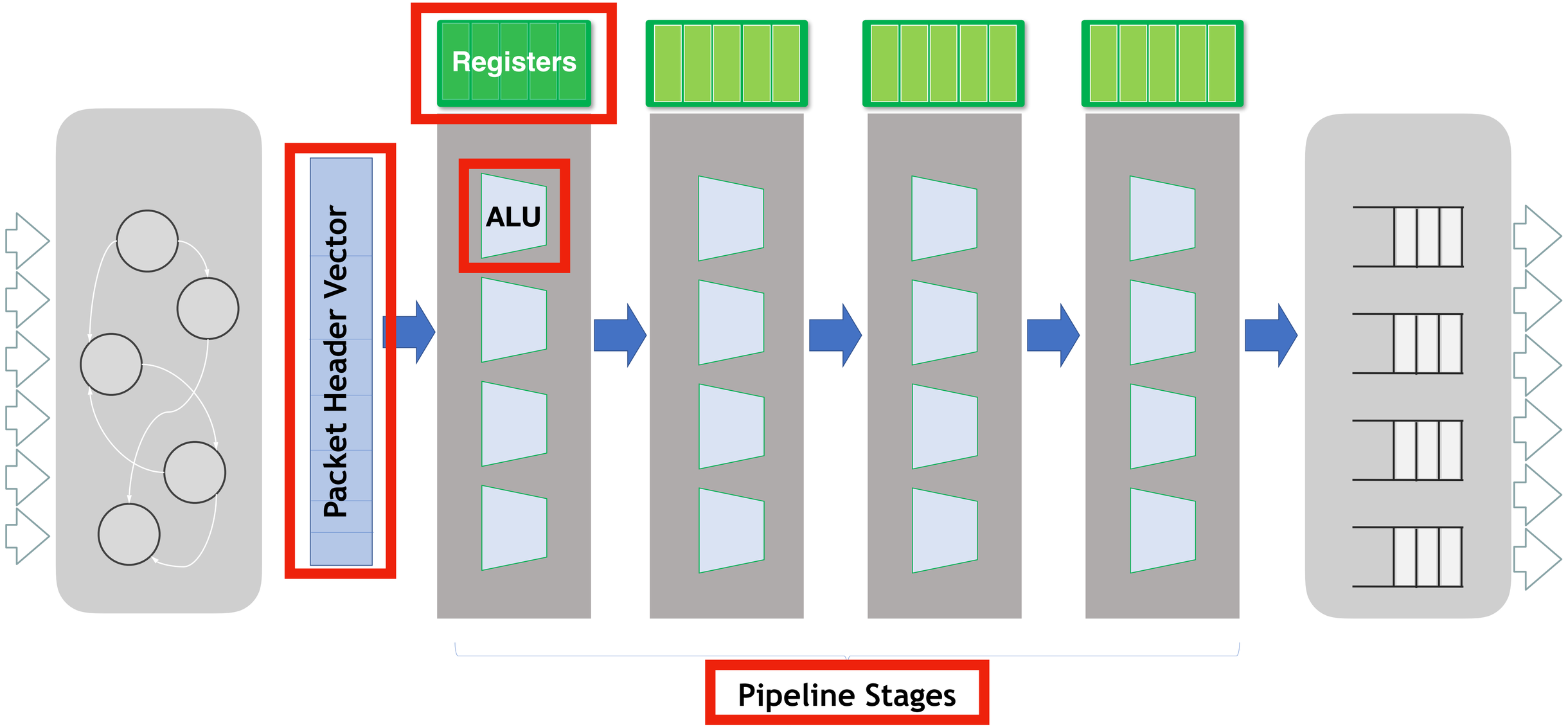
(2) Resource restrictions

Protocol-Independent Switch Architecture

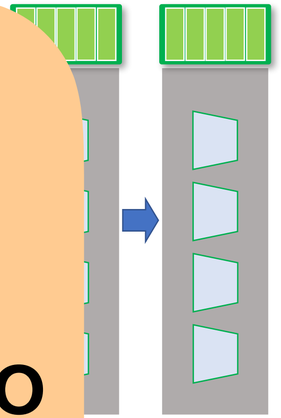
PISA


Programmable Parser
Packet Header Vector
Registers
ALU
Programmable Deparser
Pipeline Stages

PISA



Language Tightly Coupled with Hardware



The gap between expressiveness and scarcity forces programmers to make decisions about low-level details.

W
exp
prog
high

have to
y **scarce**
hand

Network Cache

Key	Value
1	A
2	B
3	C
4	D

Key-Value Store for popular content

What if the content popularity changes over time?

NetCache, *Jin et al.* [SOSP'17]

Count-Min Sketch

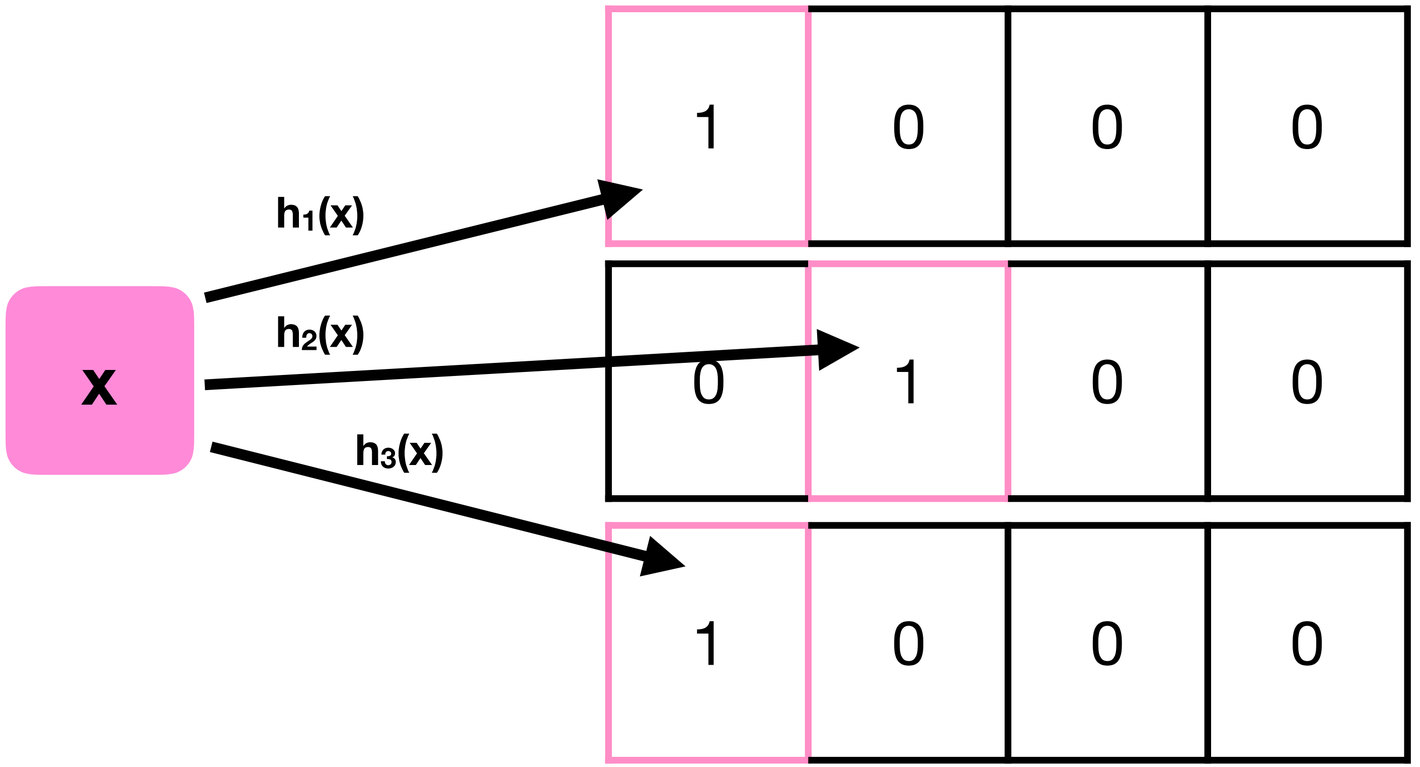
0	0	0	0
0	0	0	0
0	0	0	0

Count-Min Sketch

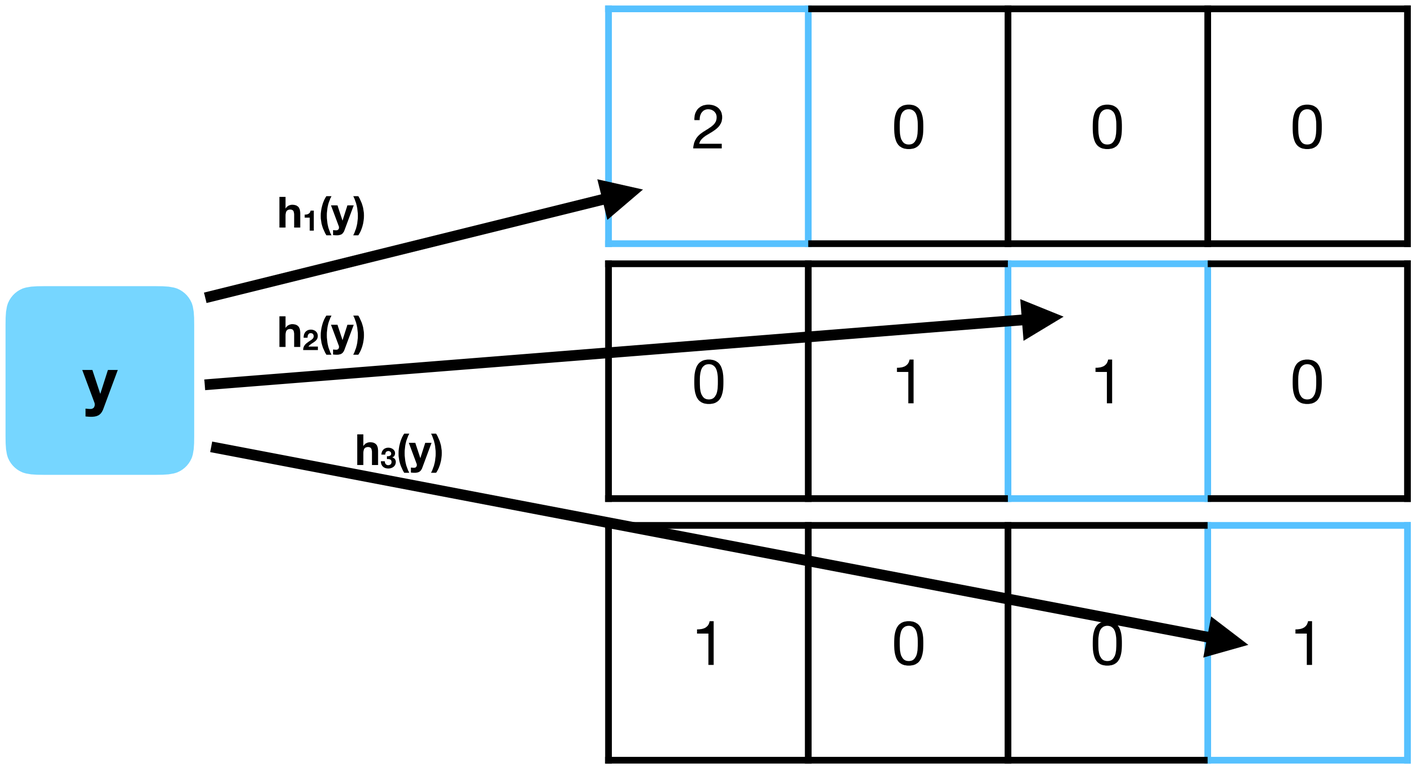
x

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

Count-Min Sketch



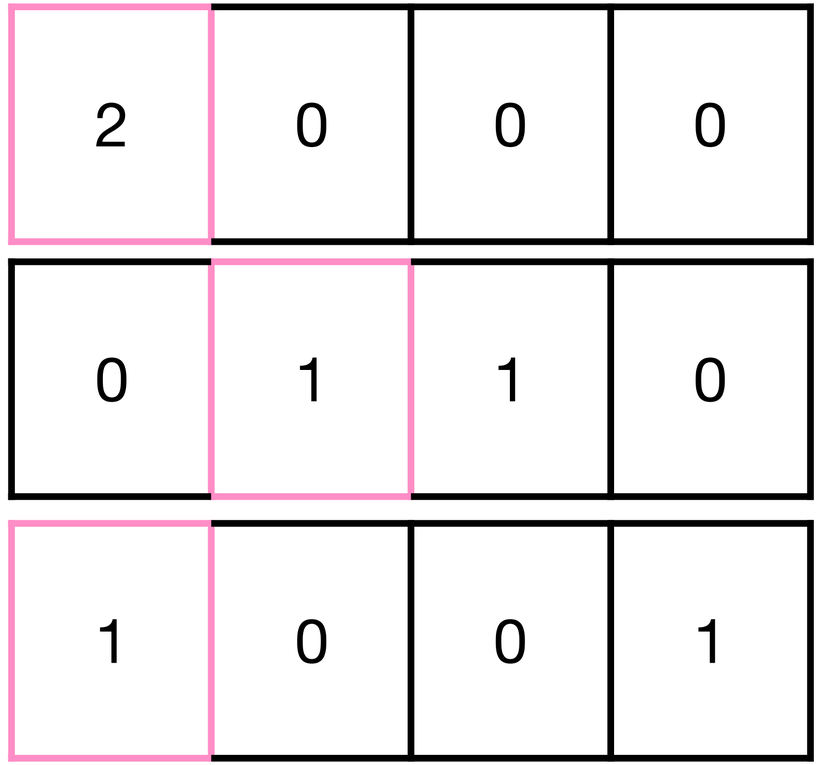
Count-Min Sketch



Count-Min Sketch

x

Count(x) = 1



Building a Cache

Key	Value
1	A
2	B
3	C
4	D

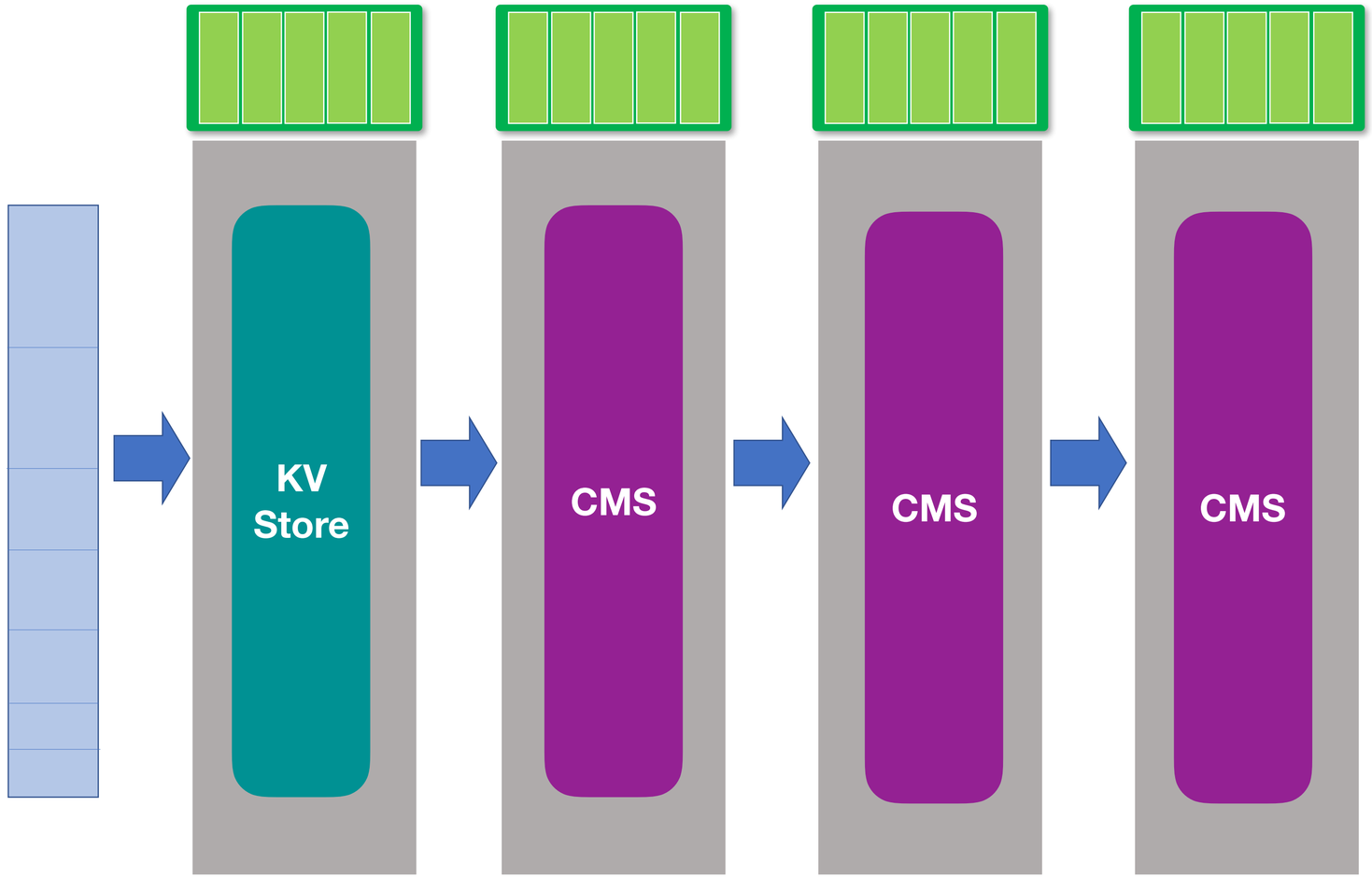
Key-Value Store for popular content

Estimated # of Requests

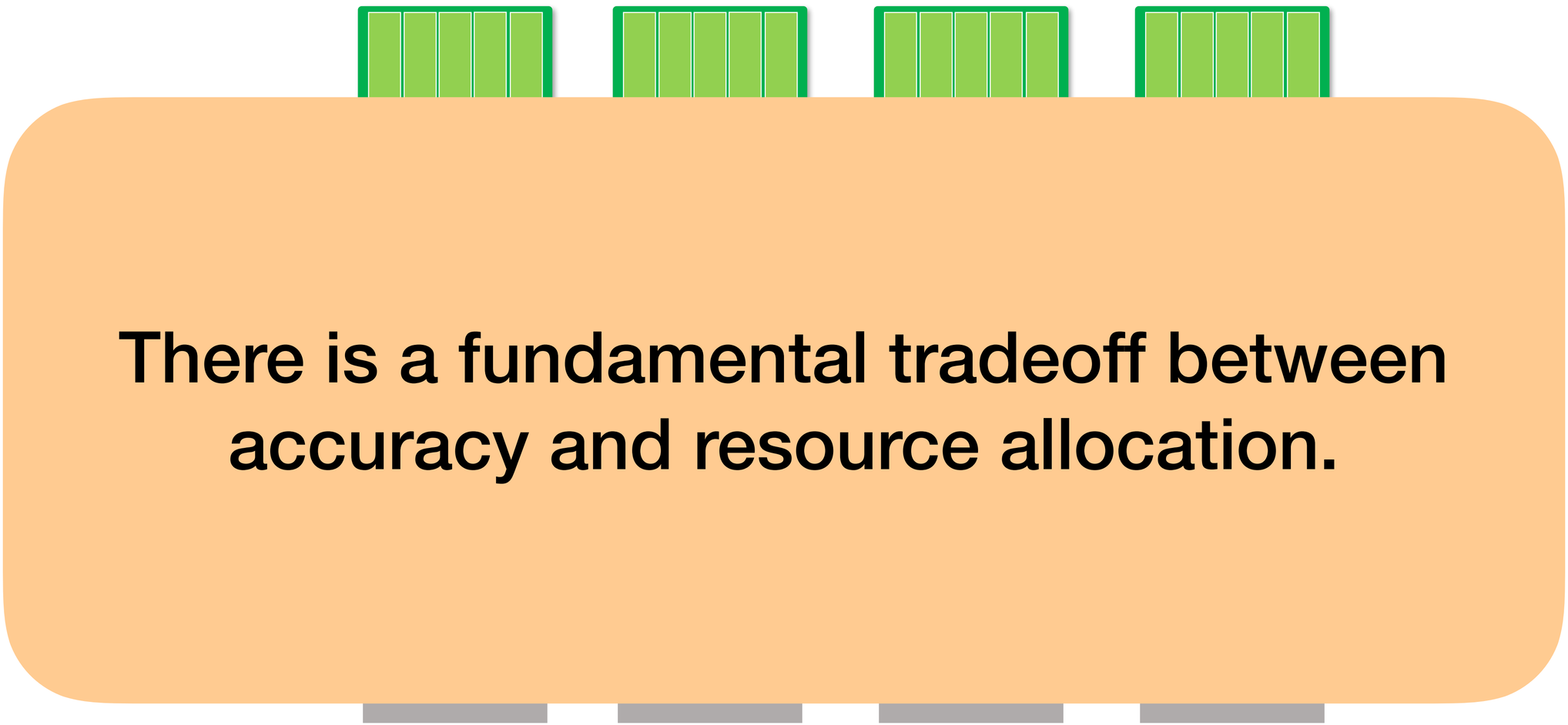
0	0	0	0
0	0	0	0
0	0	0	0

Popularity Tracker (CMS)

Allocating Resources to the Cache



Allocating Resources to the Cache



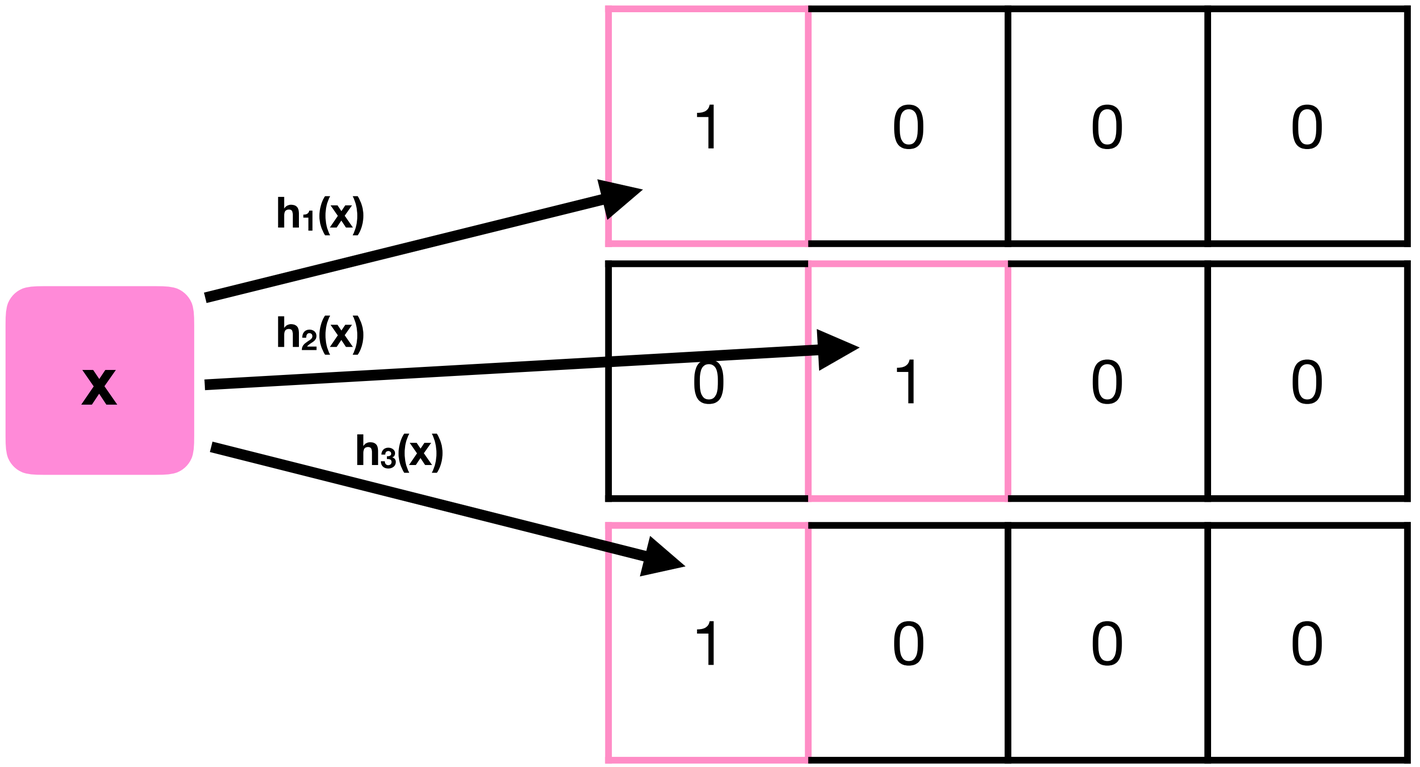
Cache in P4

```
kv_size = 100;
cms_cols = 50;
```

Cache in P4

```
kv_size = 100;
cms_cols = 50;
register<bit<32>>(cms_cols) cms_row0;
register<bit<32>>(cms_cols) cms_row1;
register<bit<32>>(cms_cols) cms_row2;
```

Count-Min Sketch



Cache in P4

```
kv_size = 100;
cms_cols = 50;
register<bit<32>>(cms_cols) cms_row0;
register<bit<32>>(cms_cols) cms_row1;
register<bit<32>>(cms_cols) cms_row2;

    def cms_hash0 () { … }
    def cms_hash1 () { … }
    def cms_hash2 () { … }
```

Low-level details are intertwined with the programming language.

The System

Practical system that decouples the programming language from the hardware to simplify programming network devices

P4All
[HotNets'20]
Parasol
[In Submission]

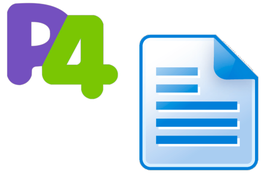
Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Automatically tailor program to environment



Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

P4All

P4All
[HotNets'20]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]

Optimization with scarce resources

Automatically tailor program to environment

Reusable **elastic structures** that can stretch or shrink to fill available resources

Elastic Switch Programming with P4All; ***Hogan****, Feibish, Arashloo, Rexford, Walker, Harrison;* HotNets'20

Cache in P4

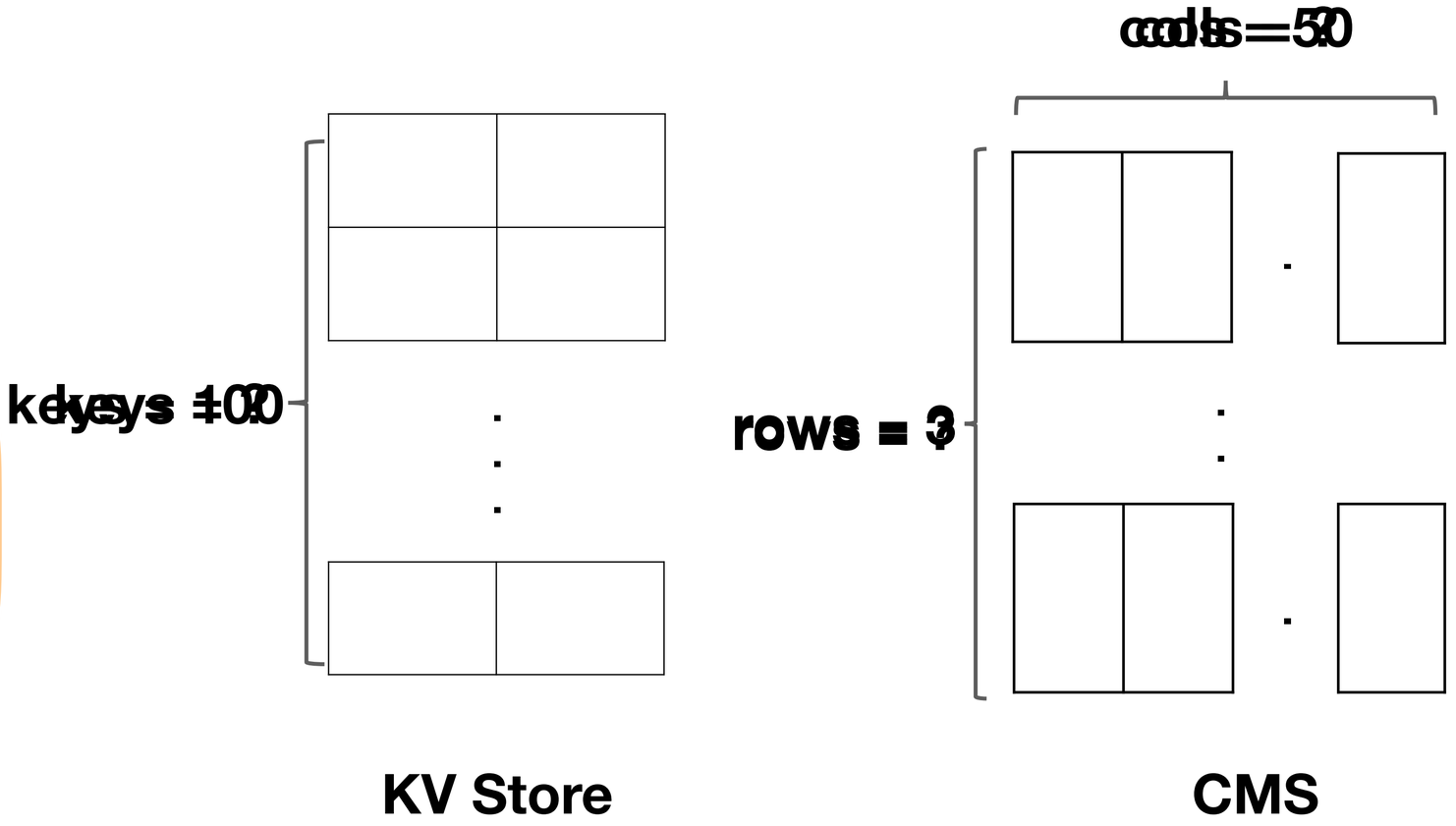
```
kv_size = 100;
cms_cols = 50;
register<bit<32>>(cms_cols) cms_row0;
register<bit<32>>(cms_cols) cms_row1;
register<bit<32>>(cms_cols) cms_row2;

    def cms_hash0 () { … }
    def cms_hash1 () { … }
    def cms_hash2 () { … }
```

Low-level details are intertwined with the programming language.

Elastic Parameters

```
symbolic int kv_size;
 kv_size = 100;
symbolic int cms_rows;
 cms_cols = 50;
symbolic int cms_cols;
```

Symbolic values are placeholders.



Elastic Switch Programming with P4All; ***Hogan****, Feibish, Arashloo, Rexford, Walker, Harrison;* HotNets'20

Elastic Structure

```
kv_size = 100;
cms_cols = 50;
register<bit<32>>(cms_cols) cms_row0;
register<bit<32>>(cms_cols) cms_row1;
register<bit<32>>(cms_cols) cms_row2;
```

P4

```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
register<bit<32>>(cms_cols)[cms_rows] cms;
```

P4All

Elastic Operations

```
kv_size = 100;
cms_cols = 50;
register<bit<32>>(cms_cols) cms_row0;
register<bit<32>>(cms_cols) cms_row1;
register<bit<32>>(cms_cols) cms_row2;

def cms_hash0 () { … }
def cms_hash1 () { … }
def cms_hash2 () { … }
```

P4

```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
register<bit<32>>(cms_cols)[cms_rows] cms;

def cms_hash () {
  for (i < cms_rows) {
    hash_and_increment(cms[i]);
  }
}
```

P4All

The System

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Elastic structures defined with symbolic values, arrays, bounded loops

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Automatically tailor program to environment

Compile to domain-specific code (P4)

P4 code that fits switch architecture

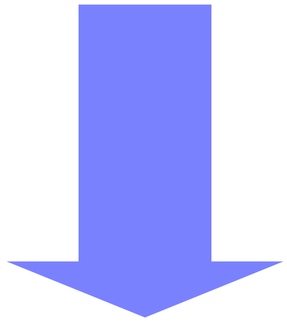
Programmable switches

Compile to hardware binary

Ready to run in the network!

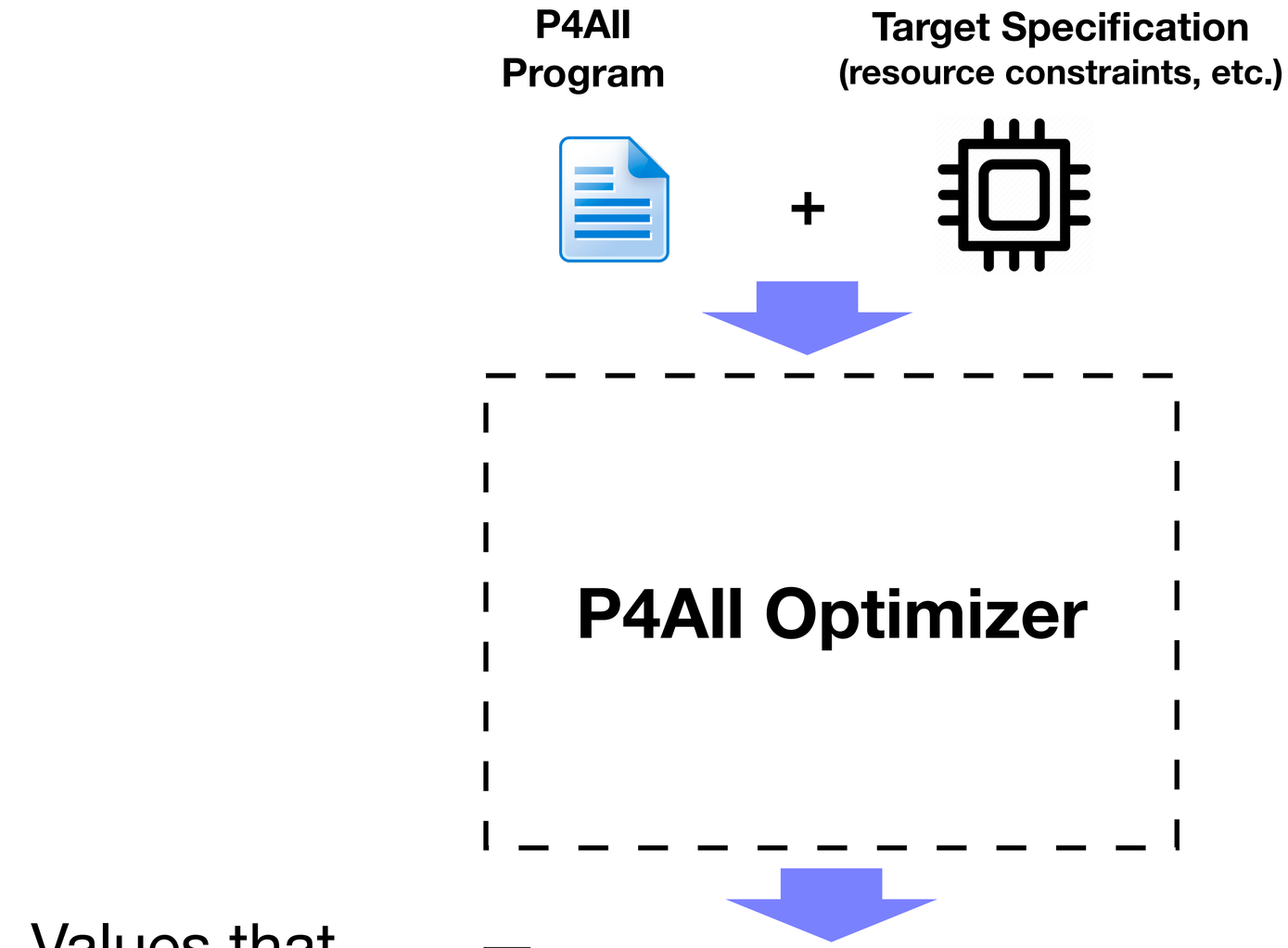
Modular Switch Programming Under Resource Constraints; ***Hogan****, Feibish, Arashloo, Rexford, Walker;* NSDI'22

P4All
Program

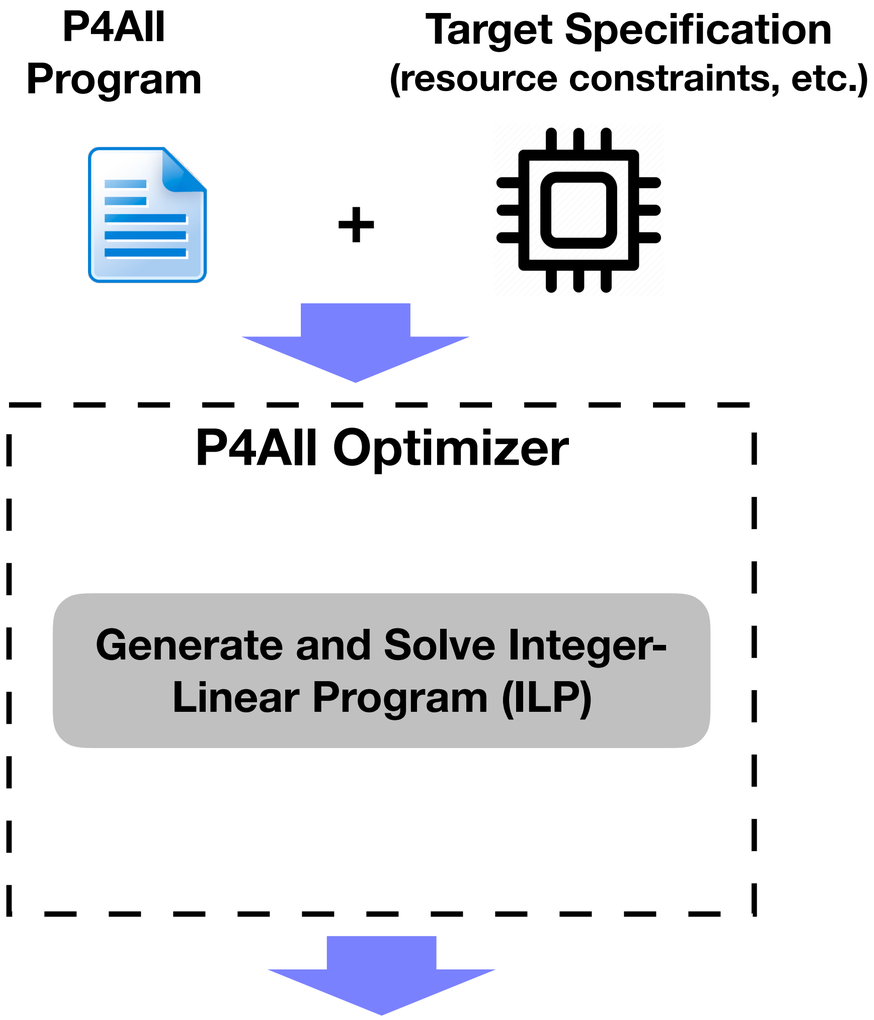


Concrete values
for symbolic values
(P4 Program)

Modular Switch Programming Under Resource Constraints; ***Hogan****, Feibish, Arashloo, Rexford, Walker;* NSDI'22



Values that yield **optimal** performance



Values that yield **optimal** performance

The System

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Objective function

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

Cache Performance

Key	Value
1	A
2	B
3	C
4	D

Key-Value Store for popular content

$$\textbf{Hit rate} = kv\_size \sum_{i=1} \frac{1}{i^{\alpha}}$$

Workload-dependent parameter

Web Caching, *Breslau et al.* [INFOCOMM'99]

Cache Performance

Key	Value
1	A
2	B
3	C
4	D

Key-Value Store for popular content

Estimated # of Requests

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

Popularity Tracker (CMS)

Cache Performance

$$\texttt{cms\_cols} = 3\,(1/\epsilon)^{1/\alpha}$$

CMS error

$$\epsilon = \frac{3}{cms\_cols}$$

Estimated # of Requests

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

Popularity Tracker (CMS)

Summarizing and Mining Skewed Data Streams, *Cormode et al.* [SDM'05]

Cache Performance

Hit rate = $\sum_{i=1}^{kv\_size} \frac{1}{i}$ $-\epsilon$ = $\frac{3}{cms\_cols}$

Optimal Allocation = Optimal Performance

ILP:

maximize **key value hit rate - CMS error**

subject to **resource constraints**

$$\left(\sum_{i=1}^{kv\_size} \frac{1}{i}\right) - \frac{3}{cms\_cols}$$

(If the objective is not linear, we use a linear approximation)

Optimal Allocation = Optimal Performance

ILP:

maximize **objective function**

subject to **resource constraints**

Objective function = performance as a function of structure size

The System

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Objective function

Generate and solve ILP

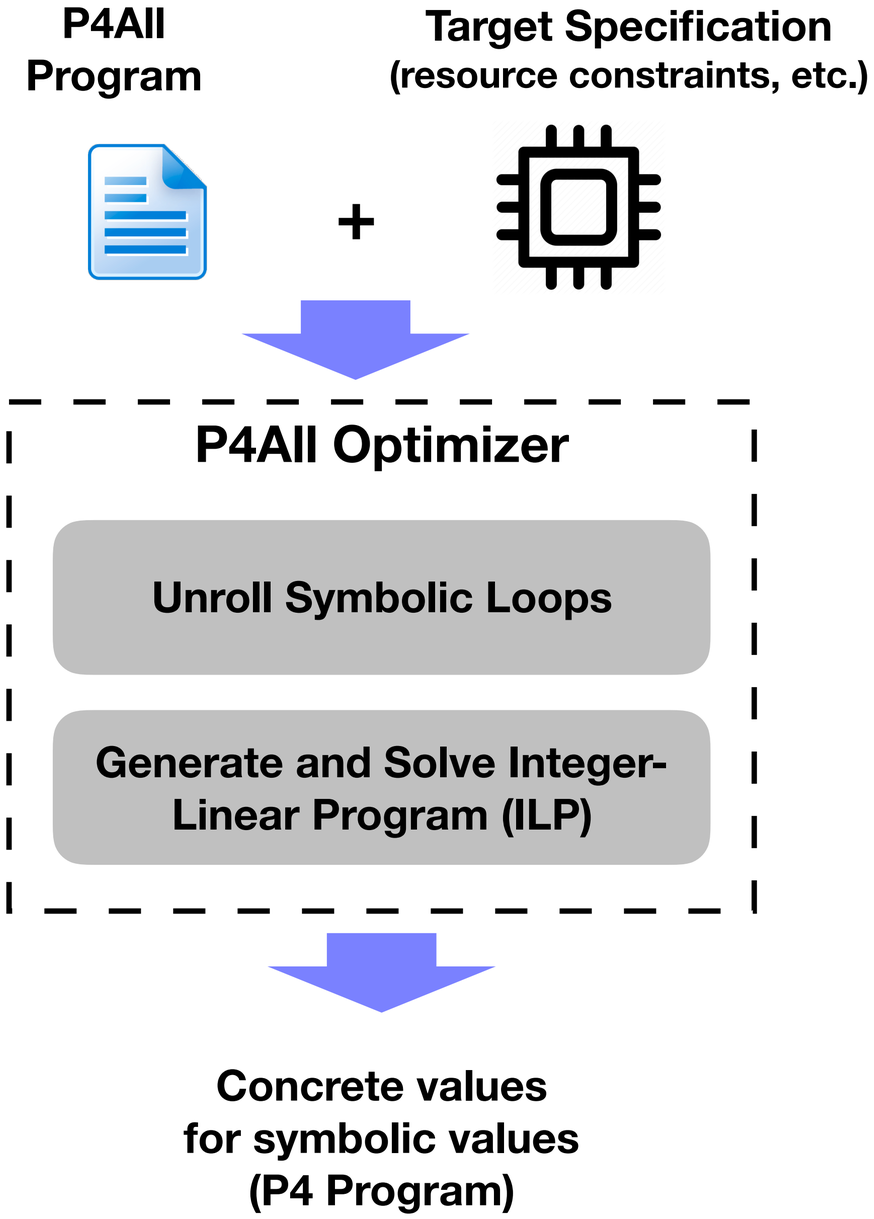
Compile to domain-specific code (P4)

P4 code that fits switch architecture

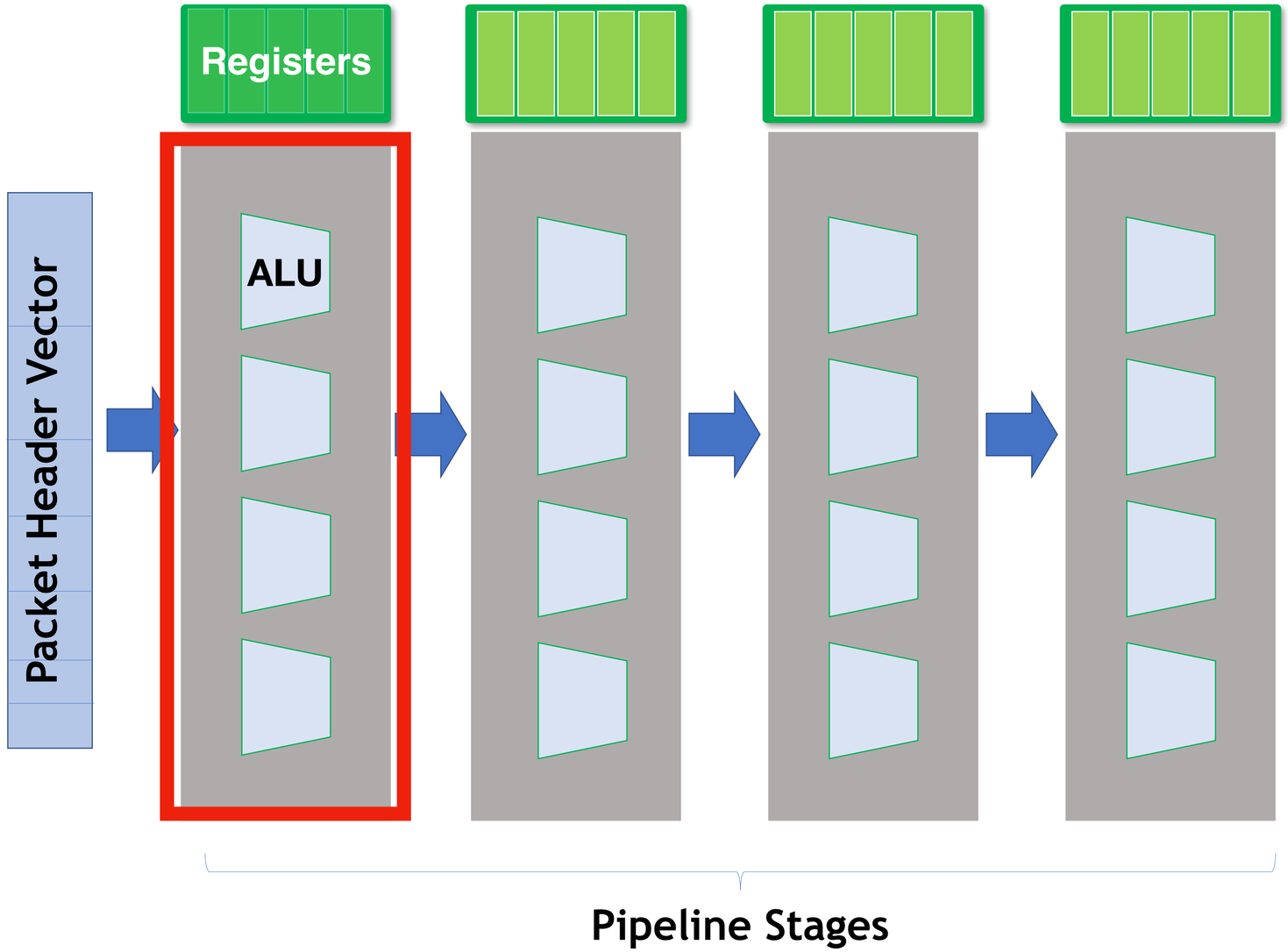
Programmable switches

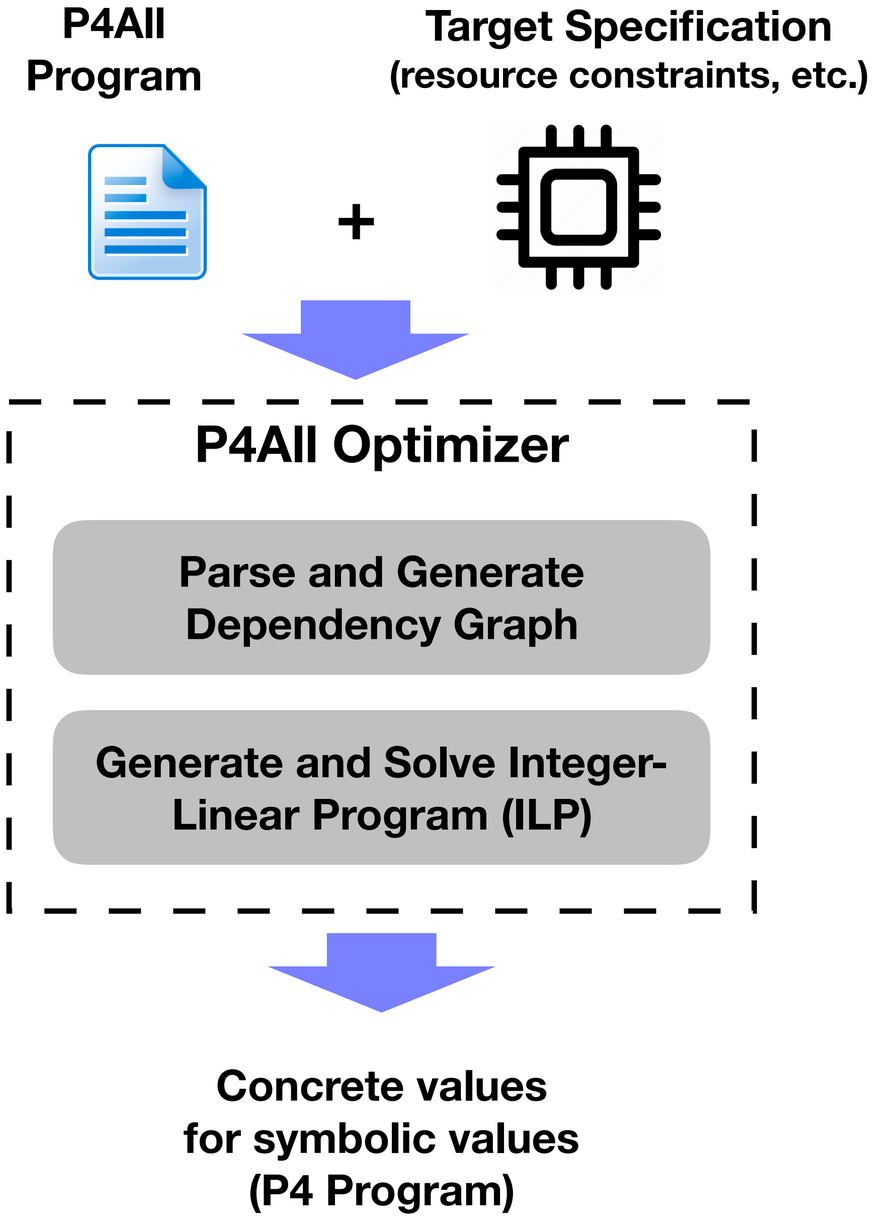
Compile to hardware binary

Ready to run in the network!


P4All
Program
Target Specification
(resource constraints, etc.)
+
P4All Optimizer
Unroll Symbolic Loops
Generate and Solve Integer-
Linear Program (ILP)
Concrete values
for symbolic values
(P4 Program)

Resources Determine Bound




P4All
Program
Target Specification
(resource constraints, etc.)
+
P4All Optimizer
Parse and Generate
Dependency Graph
Generate and Solve Integer-
Linear Program (ILP)
Concrete values
for symbolic values
(P4 Program)

Program Dependencies

```
def cms_hash () {
  for (i < cms_rows) {
    hash_and_increment(cms[i]);
  }
}
```

Read after write dependency

We **read** and **write** to `min` for each row

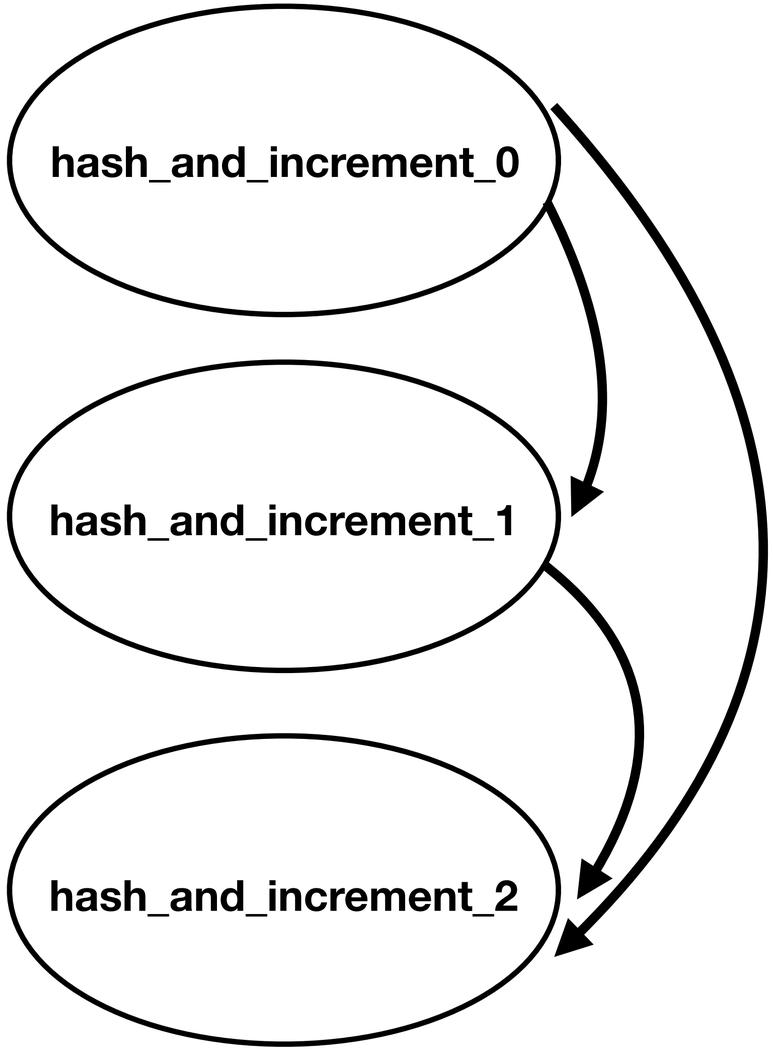
```
def hash_and_increment (cms_row) {
  int index = hash();
  int stored_val = increment(cms_row, index);
  if (stored_val < min) {
    min = stored_val; }
}
```

Dependency Graph

```
def cms_hash () {
  for (i < cms_rows) {
    hash_and_increment(cms[i]);
  }
}
```

Walk the graph until

nodes visited = number of available stages

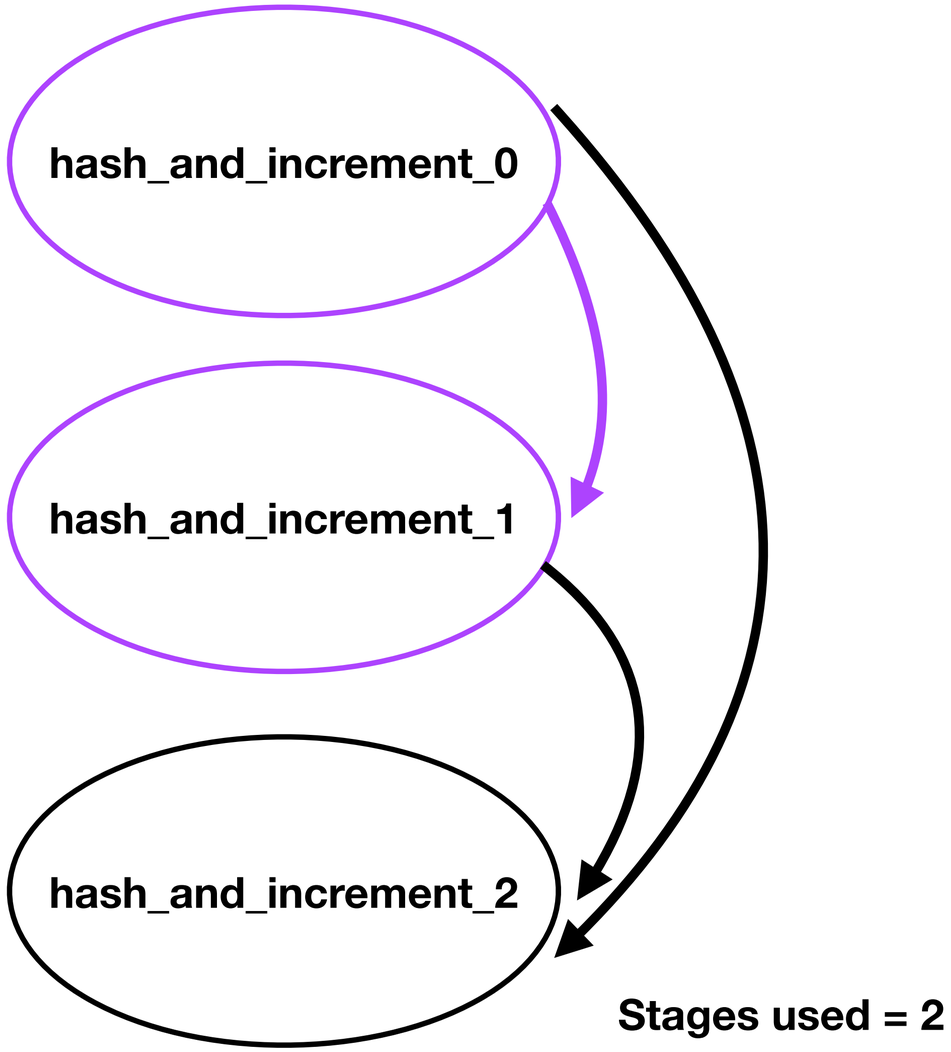


Stages available = 2

Dependency Graph

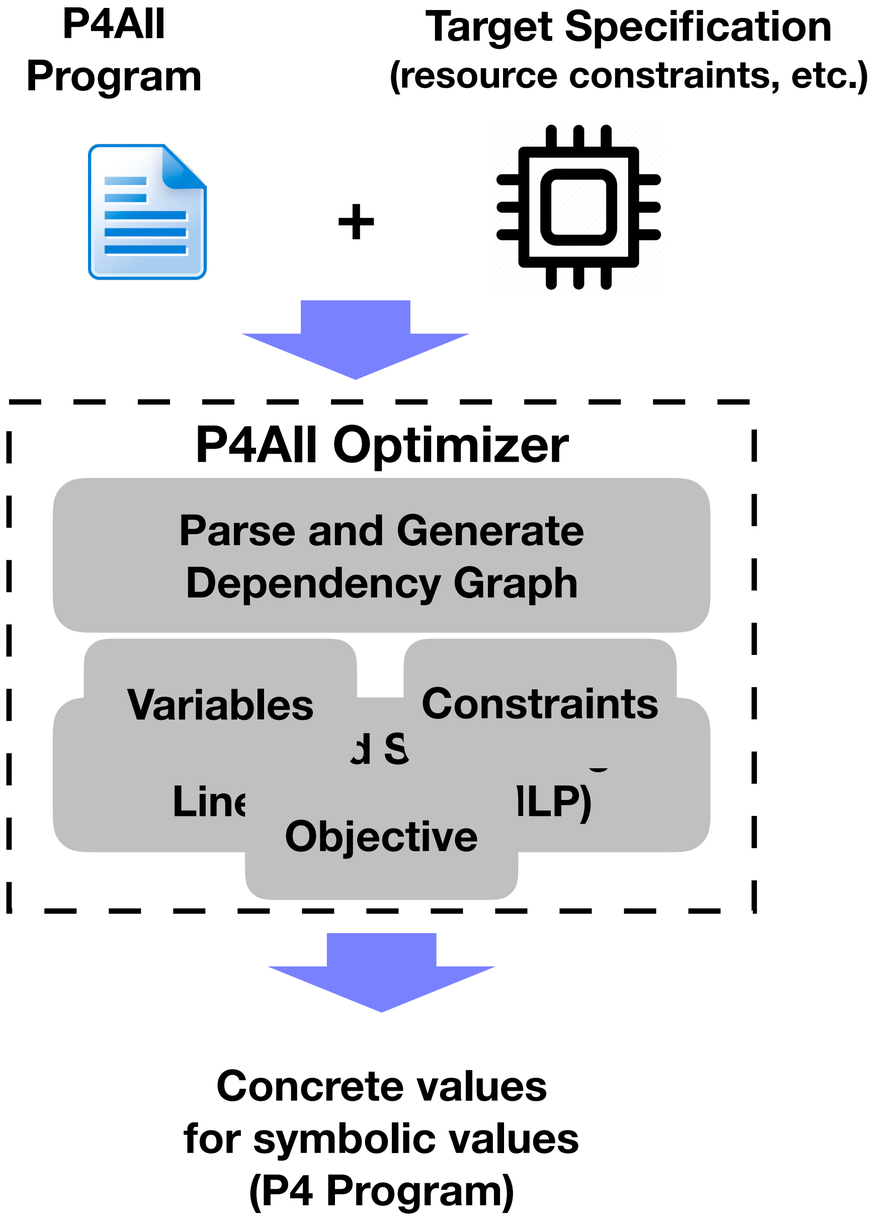
```
def cms_hash () {
  for (i < cms_rows) {
    hash_and_increment(cms[i]);
  }
}
```

cms_rows <= 2

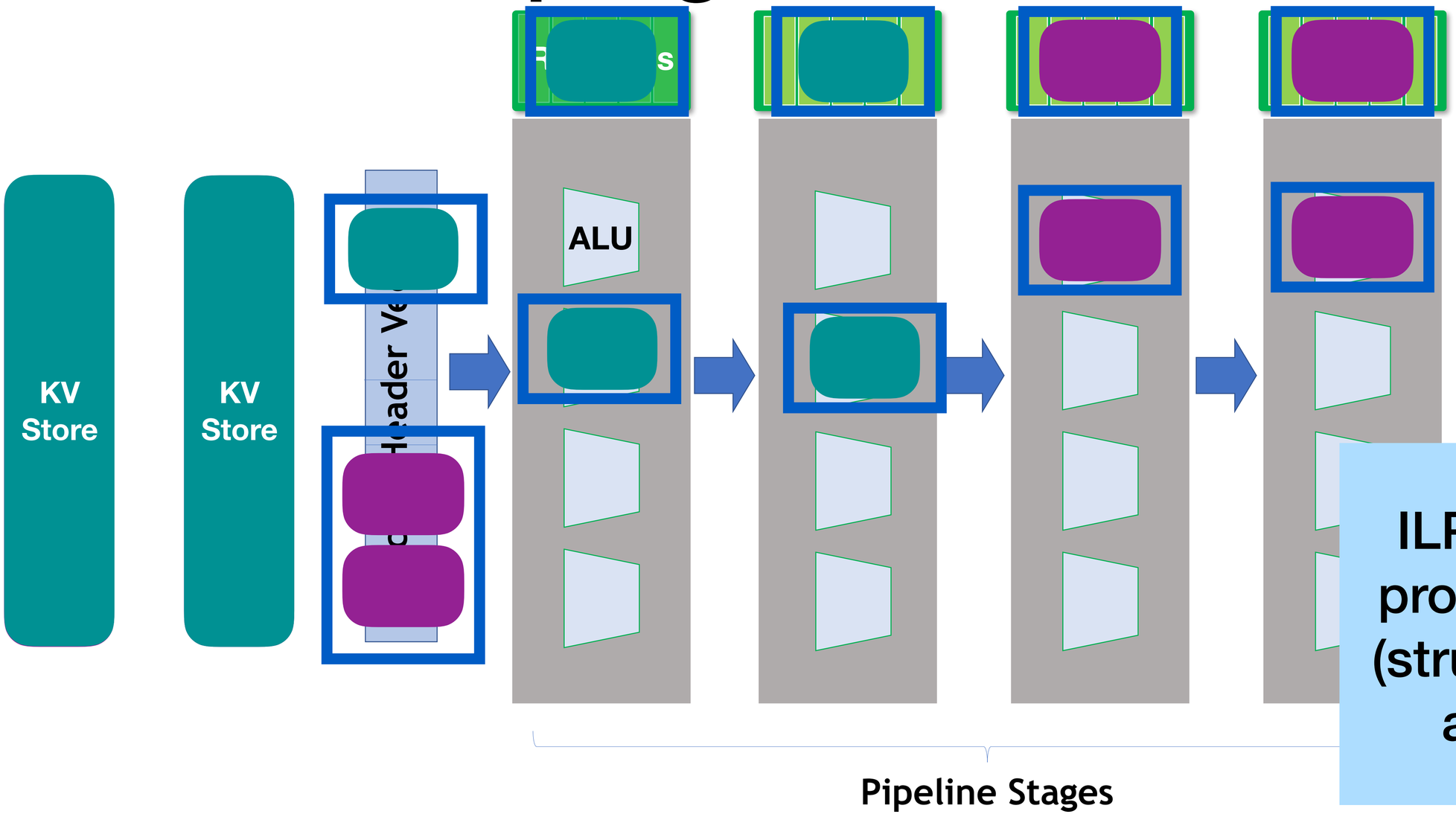


Stages available = 2

Stages used = 2

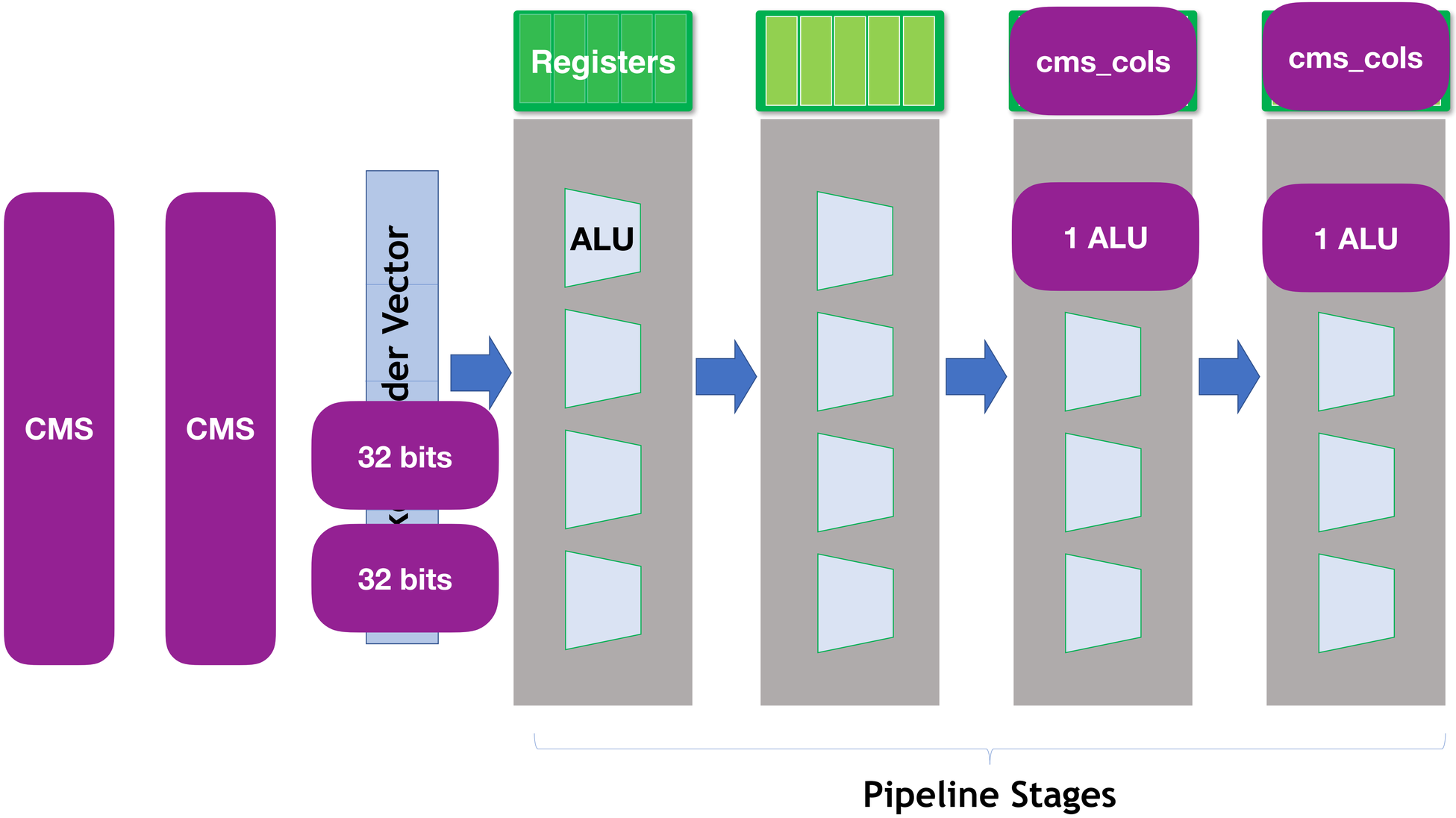

P4All
Program
Target Specification
(resource constraints, etc.)
+
P4All Optimizer
Parse and Generate
Dependency Graph
Variables
Constraints
Objective
Concrete values
for symbolic values
(P4 Program)

ILP assigns resources to program elements

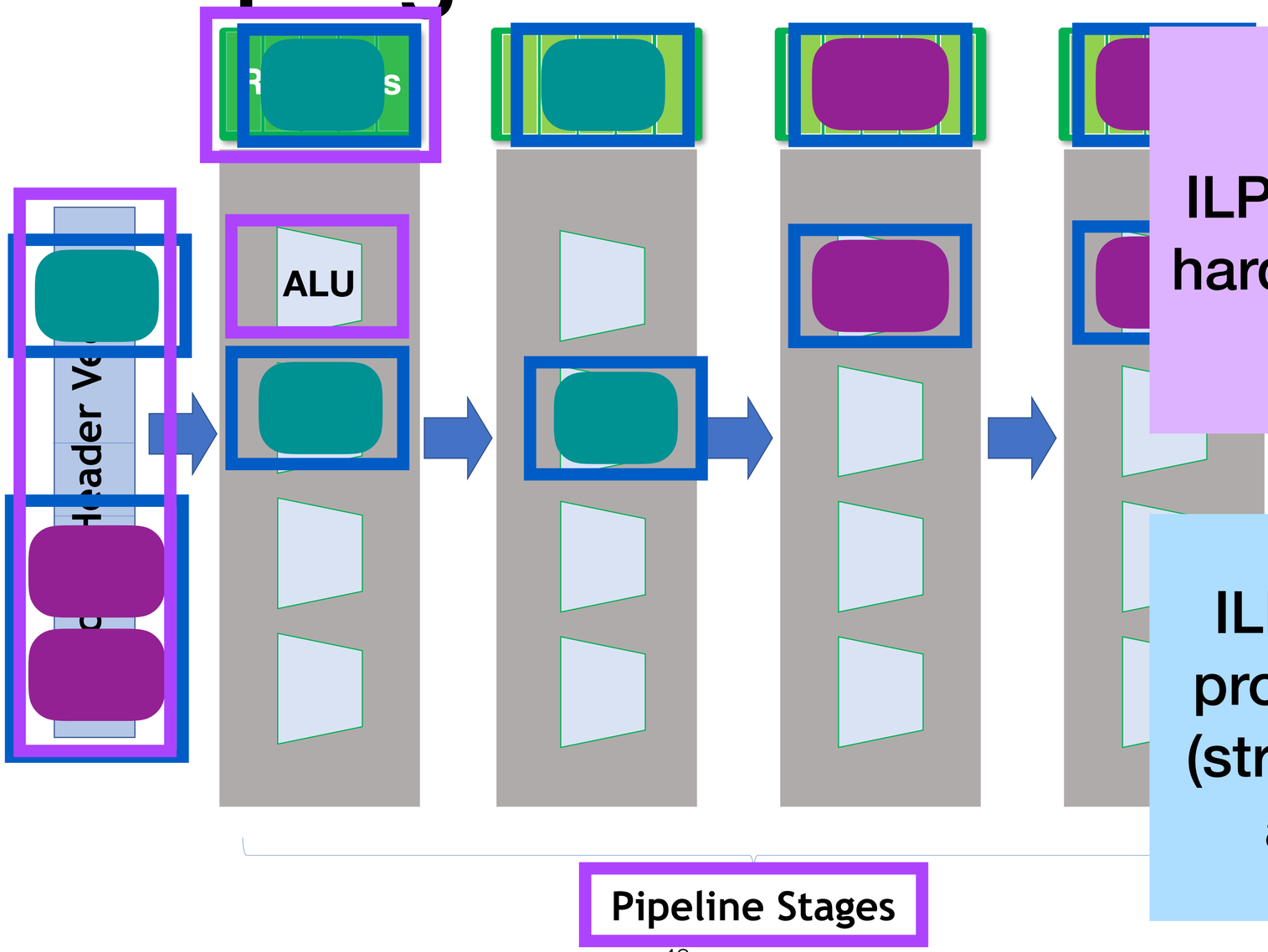


ILP variables are program elements (structure memory, actions, etc.)

ILP Variables



ILP assigns resources to program elements



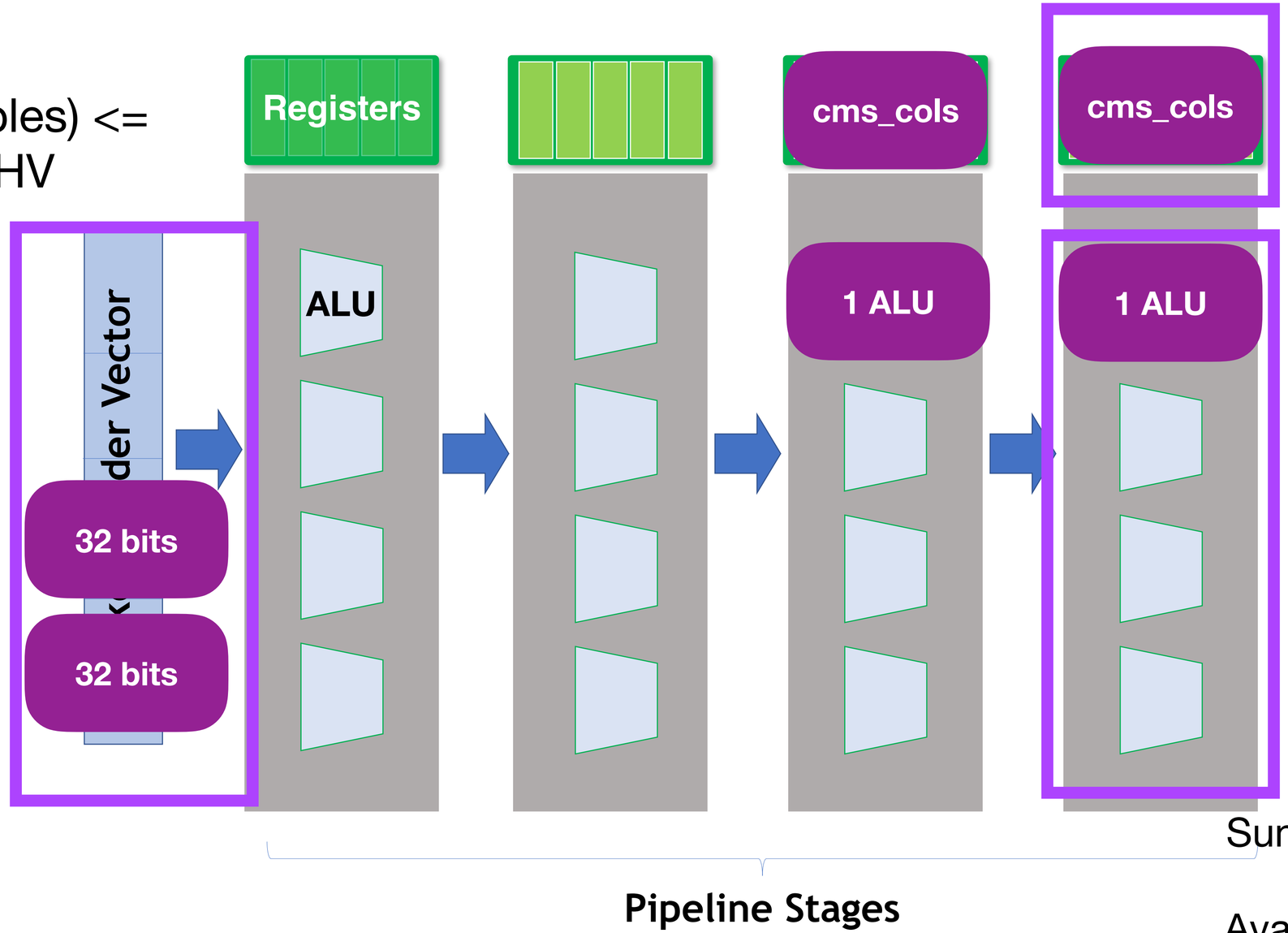
ILP constraints are hardware resources

ILP variables are program elements (structure memory, actions, etc.)

ILP Constraints

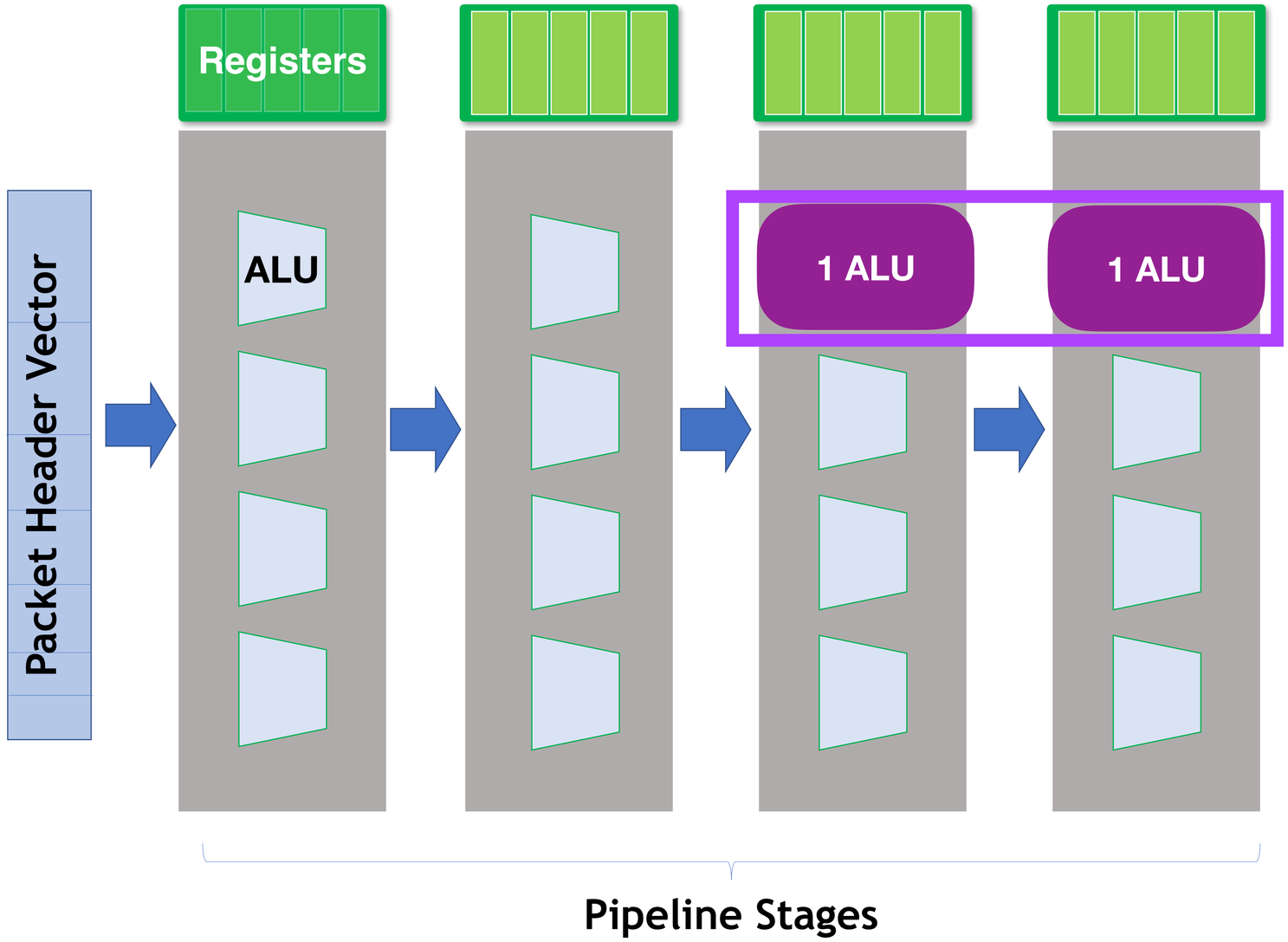
Sum(register variables per stage) <= Available registers per stage

Sum(PHV variables) <= Available PHV



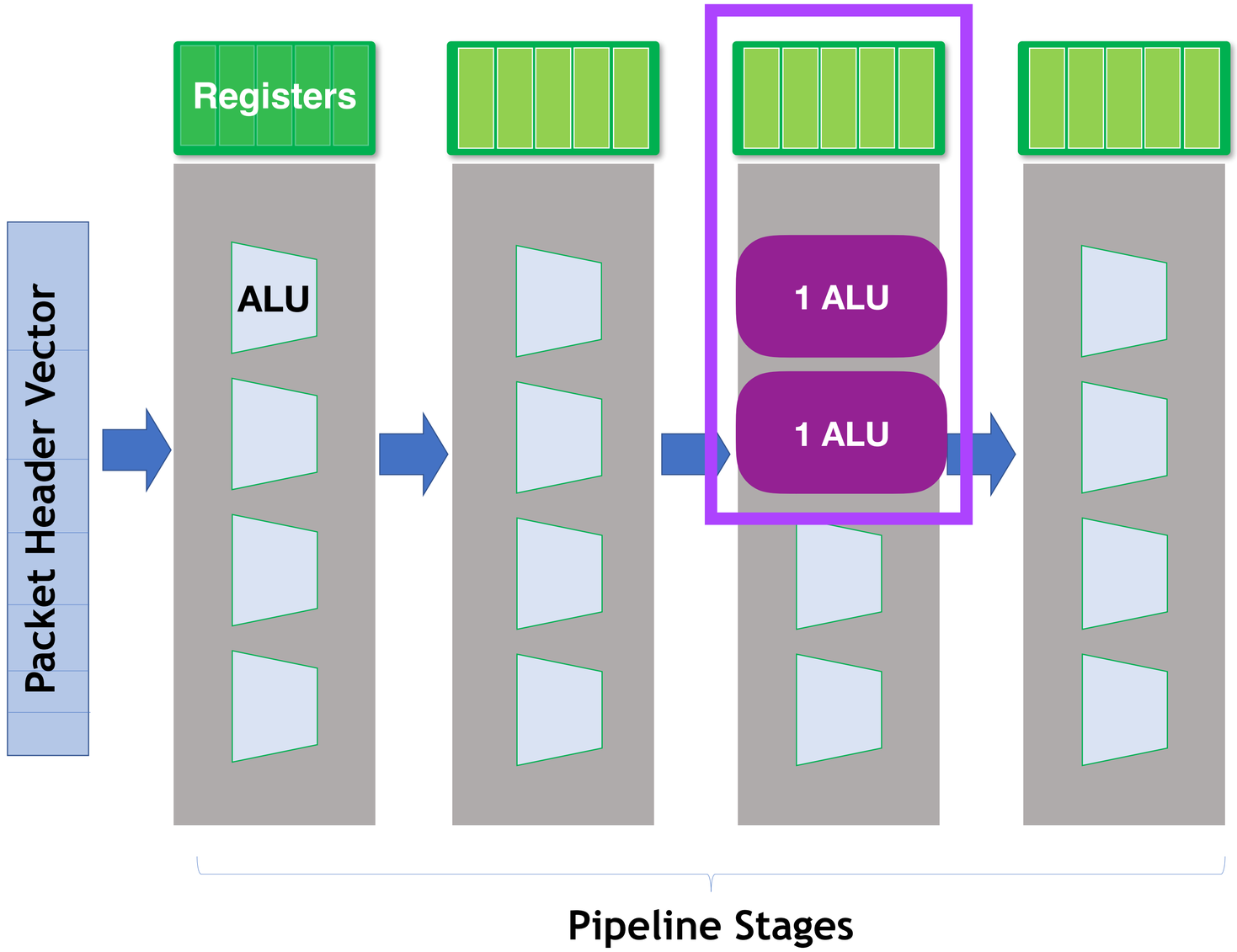
Sum(action variables per stage) <= Available ALUs per stage

ILP Constraints



If $action_0$ and $action_1$ are dependent,
$Stage_{a0}$!= $Stage_{a1}$

ILP Constraints



If $action_0$ and $action_1$ are dependent,
$Stage_{a0}$!= $Stage_{a1}$

If $action_0$ and $action_1$ access the same register arrays,
$Stage_{a0}$ = $Stage_{a1}$

Evaluation

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Elastic structures defined by symbolic values

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

How efficient/effective is the P4All optimizer?

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

Optimizer Efficiency

Application	Optimizer Time (s)
CMS	
Key-value store	
Key-value store + CMS	
Switch.p4	
IP forwarding + stateful firewall	
Beaucoup	
Precision	
NetChain	
SketchLearn	
Conquest	
Bloom filter	

Optimizer Efficiency

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

Optimizer Efficiency

Application	Optimizer 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
Bloom filter	513.6

P4All vs Hand-Optimized Code

Application
CMS
Key-value store
Key-value store + CMS
Switch.p4
IP forwarding + stateful firewall
Beaucoup
Precision
NetChain
SketchLearn
Conquest
Bloom filter

P4All generates same resource usage as hand-optimized code

P4All solution achieves marginally higher accuracy than hand-optimized code

P4All solution requires more resources

Does P4All Generalize?

P4All
[NSDI'22]

Objective function = performance as a function of structure size

Analytical functions provide bounds on worst-case performance

Parameters only relate to resource usage

ILP optimizes resource usage

Workload dependence information is limited

P4All works well when we have an analytical function and are only concerned with resource usage

Analytical functions not always possible

Optimization Objective

P4All
[NSDI'22]

Objective function = performance as a function of structure size

Parasol
[In Submission]

Objective function = run-time performance measurements

Parasol

Parasol
[In Submission]

Language abstraction for expressiveness

Separate how to evaluate from any program code

Parasol
[In Submission]

Optimization with scarce resources

Automatically tailor program to environment

Simulation-based optimization finds parameter values for optimal performance

Optimize **any** program parameter using empirical performance objectives

Automated Optimization of Parameterized Data-Plane Programs; ***Hogan***, *Loehr, Sonchack, Feibish, Rexford, Walker*

Extended Symbolics

Objective (P4All) = $\sum_{i=1}^{kv\_size} \frac{1}{i}$ - $\frac{3}{cms\_cols}$

Cache.p4

```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
def cms_hash () {
  for (i < cms_rows) {
    hash_to_row(i);
  } }
```

When should cache entries expire?

When is a key popular enough to go into the cache?

Is CMS the right data structure?

Non-resource parameters

P4All

Extended Symbolics

Objective (Parasol) = $\frac{\text{number of cache hits}}{\text{number of cache requests}}$

When should cache entries expire?

When is a key popular enough to go into the cache?

Is CMS the right data structure?

Cache.p4

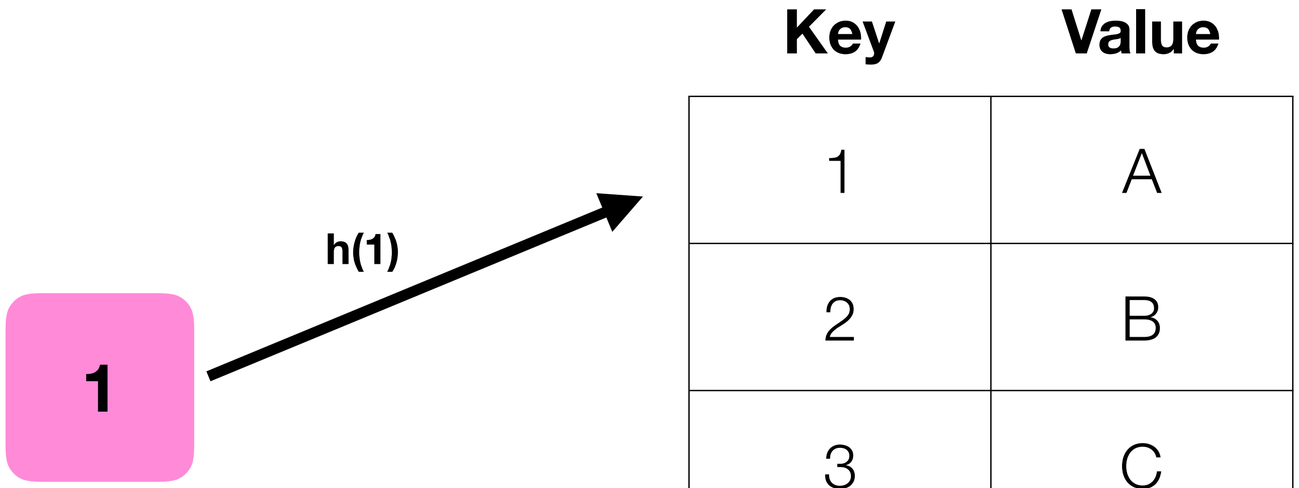
```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
def cms_hash () {
  for (i < cms_rows) {
    hash_to_row(i);
  } }
```

Cache.p4 (Cache.dpt)

```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
symbolic int timeout;
symbolic int threshold;
def cms_hash () {
  for (i < cms_rows) {
    hash_to_row(i);
  } }
```

P4All Lucid, *Sonchack et al.* [SIGCOMM'21] **Parasol**

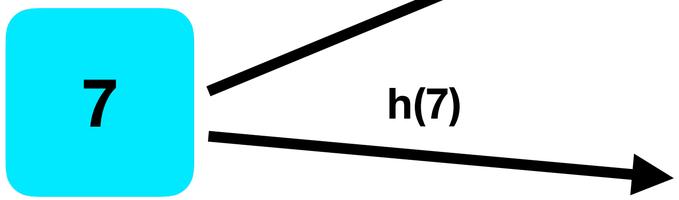
Precision Cache



Key	Value
1	A
2	B
3	C
4	D
5	E
6	F

Efficient Measurement on Programmable Switches, *Basat et al.* [ICNP'18]

Precision Cache



Key	Value	Request Count
1	A	11
2	B	50
3	C	20
4	D	100
5	E	80
6	F	5

With some probability ***p***, we evict the item with the smallest count and add the new key.

Efficient Measurement on Programmable Switches, *Basat et al.* [ICNP'18]

Choice of Data Structure

Is CMS the right data structure?

Cache.p4

```
symbolic int kv_size;
symbolic int tracker_rows;
symbolic int tracker_cols;
symbolic int timeout;
symbolic int threshold;
symbolic bool cms_tracker;

module CMS {...}
module Precision {...}
tracker = CMS if cms_tracker else Precision;
```

The System

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Extend symbolic values to *any* parameter

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Simulate and measure performance

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

Objective Functions

$$\text{Hit rate} = \frac{\text{number of cache hits}}{\text{number of cache requests}}$$

Extern Functions

$$\text{Hit rate} = \frac{\text{number of cache hits}}{\text{number of cache requests}}$$

Cache.p4

```
if (key in cache) {
    LogCacheHit();
}
LogCacheRequest();
```

Externs are like “ghost code” that run ONLY during simulation

Extern Functions

$$\text{Hit rate} = \frac{\text{number of cache hits}}{\text{number of cache requests}}$$

Cache.p4

```
if (key in cache) {
    LogCacheHit();
}
LogCacheRequest();
```

Measurements.py

```
requests = 0
hits = 0
def LogCacheHit():
    hits += 1
def LogCacheRequest():
    requests += 1
```

Scoring Performance

Cache.p4

```
if (key in cache) {
    LogCacheHit();
}
LogCacheRequest();
```

HitRate.py

```
def CacheScore(hits, requests):
    return hits / requests
```

Measurements.py

```
requests = 0
hits = 0
def LogCacheHit():
    hits += 1
def LogCacheRequest():
    requests += 1
```

vs

$$\sum_{i=1}^{kv\_size} \frac{1}{i^{\alpha}}$$

The System

P4All
[HotNets'20]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]

Parasol
[In Submission]

Optimization with scarce resources

Choose parameters to simulate

Simulate and measure performance

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

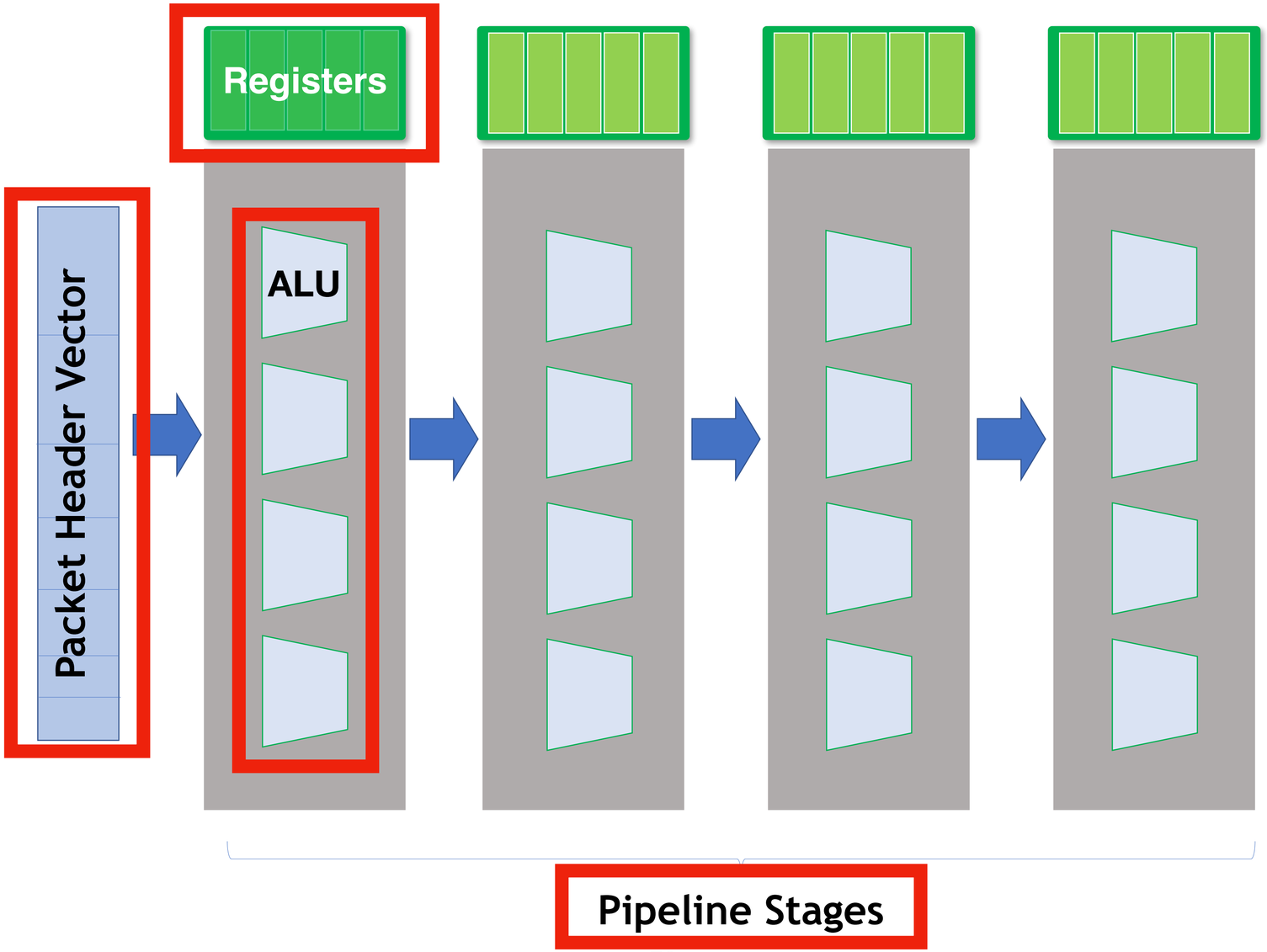
Compile to hardware binary

Ready to run in the network!

Choosing Parameter Values

```
symbolic int kv_size;
symbolic int tracker_rows;
symbolic int tracker_cols;
symbolic int timeout;
symbolic int threshold;
symbolic bool cms_tracker;
```

Resource Constraints

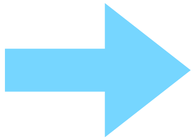


Resources Limit Parameter Values

```
symbolic int kv_size;
symbolic int tracker_rows;
symbolic int tracker_cols;
```

We use heuristics to estimate resource usage

Compilation fails if program uses too many resources



We can compile* the program (with chosen allocation) to see if it fits within resource constraints.

We get upper bounds on all resource-related symbolic values.

The System

P4All
[HotNets'20]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Remove invalid parameter choices

Choose parameters to simulate

Simulate and measure performance

Compile to domain-specific code (P4)

P4 code that fits switch architecture

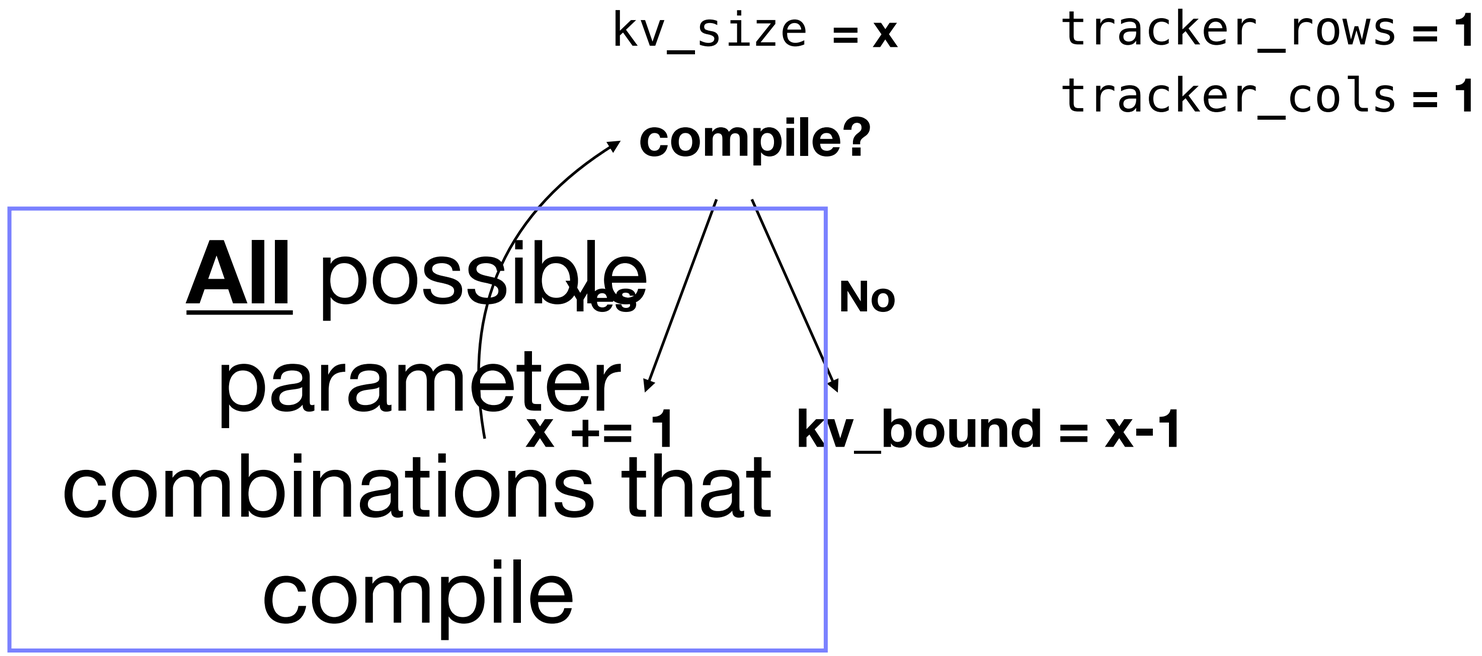
Programmable switches

Compile to hardware binary

Ready to run in the network!

Preprocessing

```
symbolic int kv_size;
symbolic int tracker_rows;
symbolic int tracker_cols;
```



kv_size : {1, ..., kv_bound} tracker_rows : {1, ..., rows_bound} tracker_cols : {1, ..., cols_bound}

The System

P4All
[HotNets'20]

Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Remove invalid parameter choices

Choose parameters to simulate

Simulate and measure performance

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

Search Strategies

(0) Exhaustive Search

(1) Nelder-Mead Simplex

(2) Simulated Annealing

(3) Bayesian Optimization

Gradient Descent

Genetic Algorithms

anything in Python!

Use the history of previous performance scores to guide the search

Parasol Overview

Cache.p4

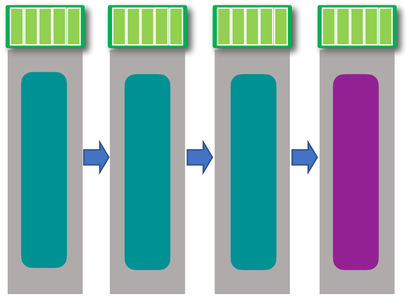
KV store, popularity tracker data structures

Remove infeasible allocations

Find data structure sizes that fit in memory

Choose parameters to simulate

Cache Requests



Simulate and measure performance

Measurements.py

Track hits, requests

HitRate.py

Pick parameters with search strategy

Choose structure, structure sizes, threshold

Simulate cache requests

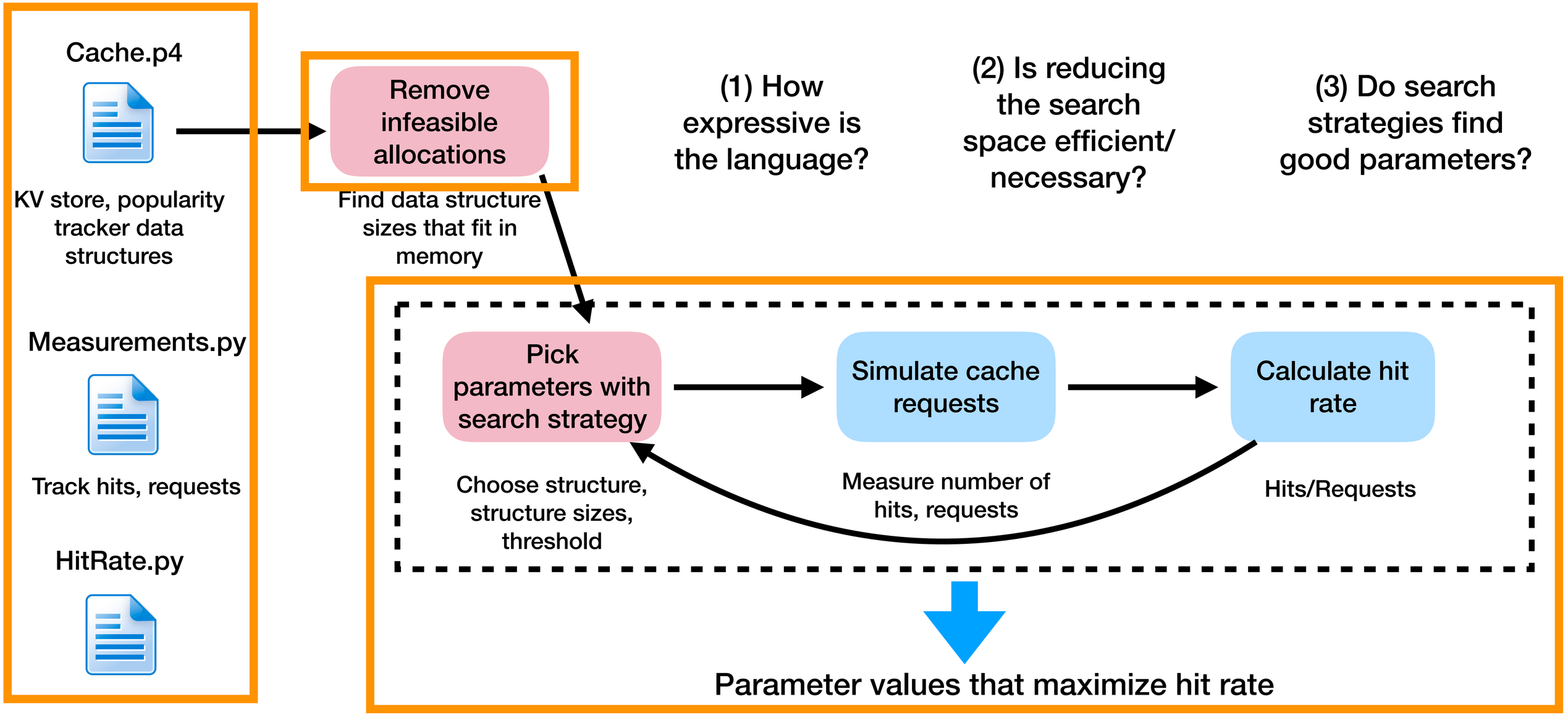
Measure number of hits, requests

Calculate hit rate

Hits/Requests

Parameter values that maximize hit rate

Parasol Evaluation



Parasol Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

P4All vs Parasol Language

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

P4All vs Parasol Language

Application	Params in P4All?
Cache	✗
Hash Table	
Stateful Firewall	✗
Load Balancer	✗
Count-Min Sketch	
Precision	
Measurement Queries	
Congestion Detection	
(Short) RTT Measurement	✗
(Long) RTT Measurement	✗

P4All vs Parasol Language

Application	Params in P4All?	Objective in P4All?
Cache	✗	✗
Hash Table		
Stateful Firewall	✗	✗
Load Balancer	✗	✗
Count-Min Sketch		
Precision		✗
Measurement Queries		✗
Congestion Detection		✗
(Short) RTT Measurement	✗	✗
(Long) RTT Measurement	✗	

Preprocessing Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

How long did it take to reduce the search space?

Preprocessing Evaluation

Application	Reduction Time
Cache	2 hours
Hash Table	15 seconds
Stateful Firewall	30 seconds
Load Balancer	2 seconds
Count-Min Sketch	16 seconds
Precision	32 minutes
Measurement Queries	1.5 hours
Congestion Detection	15 seconds
(Short) RTT Measurement	23 seconds
(Long) RTT Measurement	3 seconds

Preprocessing Evaluation

Application	Reduction Time
Cache	2 hours
Hash Table	**15 seconds**
Stateful Firewall	**30 seconds**
Load Balancer	**2 seconds**
Count-Min Sketch	**16 seconds**
Precision	32 minutes
Measurement Queries	1.5 hours
Congestion Detection	**15 seconds**
(Short) RTT Measurement	**23 seconds**
(Long) RTT Measurement	**3 seconds**

Preprocessing Evaluation

Application	Reduction Time
Cache	**2 hours**
Hash Table	15 seconds
Stateful Firewall	30 seconds
Load Balancer	2 seconds
Count-Min Sketch	16 seconds
Precision	32 minutes
Measurement Queries	**1.5 hours**
Congestion Detection	15 seconds
(Short) RTT Measurement	23 seconds
(Long) RTT Measurement	3 seconds

Preprocessing Evaluation

Optimizer found better solutions with reduced search space!

Application	Reduction Time
Cache	**2 hours**
Hash Table	15 seconds
Stateful Firewall	**30 seconds**
Load Balancer	2 seconds
Count-Min Sketch	16 seconds
Precision	32 minutes
Measurement Queries	**1.5 hours**
Congestion Detection	**15 seconds**
(Short) RTT Measurement	23 seconds
(Long) RTT Measurement	3 seconds

Search Strategy Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

Do search strategies find parameter values comparable to hand-optimized code?

Parasol vs Hand-Optimized Code

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

Parasol programs had (nearly) identical performance to hand-optimized code

P4All vs Parasol

```
symbolic int kv_size;
symbolic int cms_rows;
symbolic int cms_cols;
symbolic int timeout;
symbolic int threshold;
```

P4All and Parasol generate the same memory allocation

Extra 100k packets forwarded to storage server!

Parasol threshold (300) → **Miss rate = 66%**

Suboptimal threshold (5000) → **Miss rate = 68%**

P4All vs Parasol

```
symbolic int kv_size;
symbolic int tracker_rows;
symbolic int tracker_cols;
symbolic int timeout;
symbolic int threshold;
symbolic bool cms_tracker;
```

Distribution	CMS Miss Rate	Precision Miss Rate
High skew		
Moderate skew		

Summary

Bridge the gap between expressiveness needed for complex programs and resource scarcity

P4All
[HotNets'20]
Parasol
[In Submission]

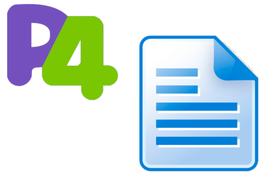
Language abstraction for expressiveness

Separate low-level details from program code

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Automatically tailor program to environment



Compile to domain-specific code (p4)

P4 code that fits switch architecture

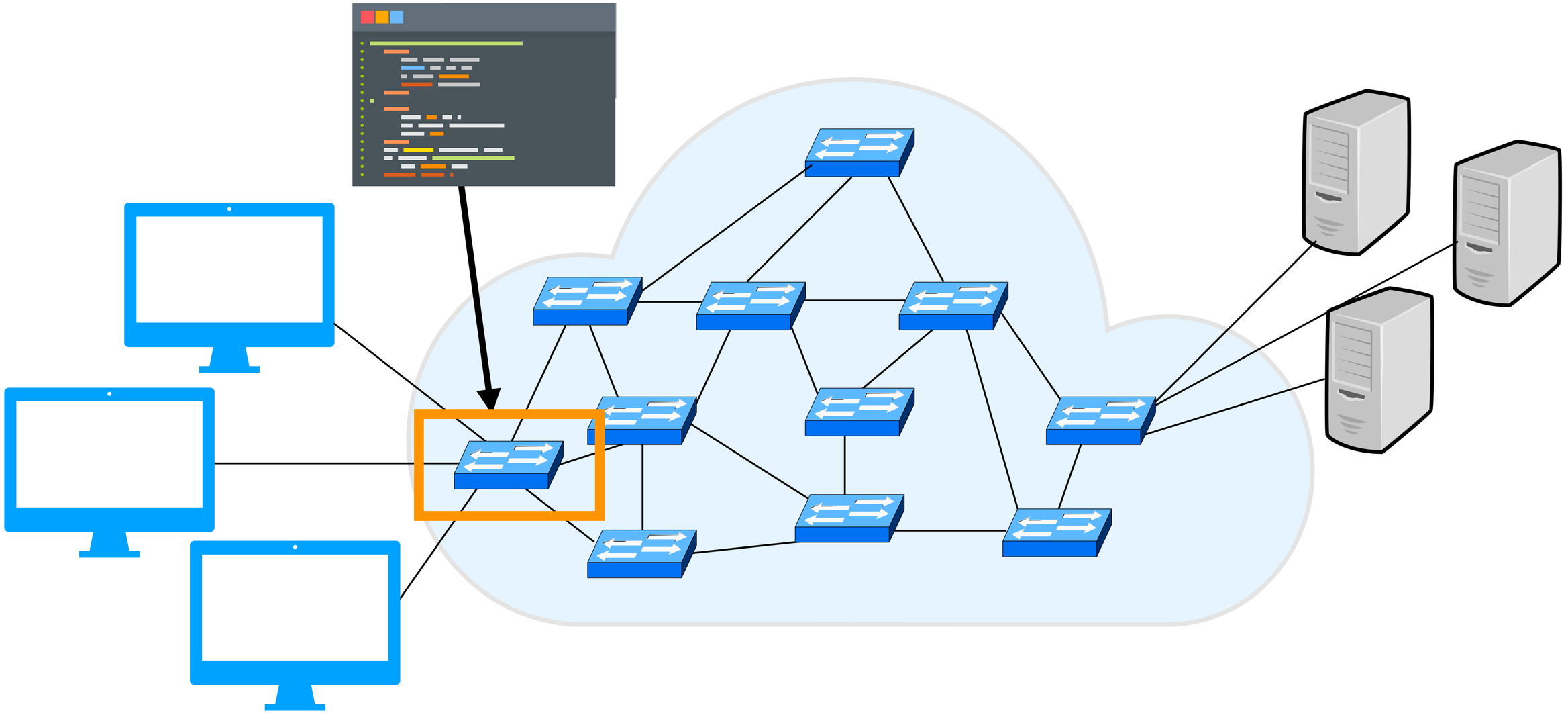
Programmable switches

Compile to hardware binary

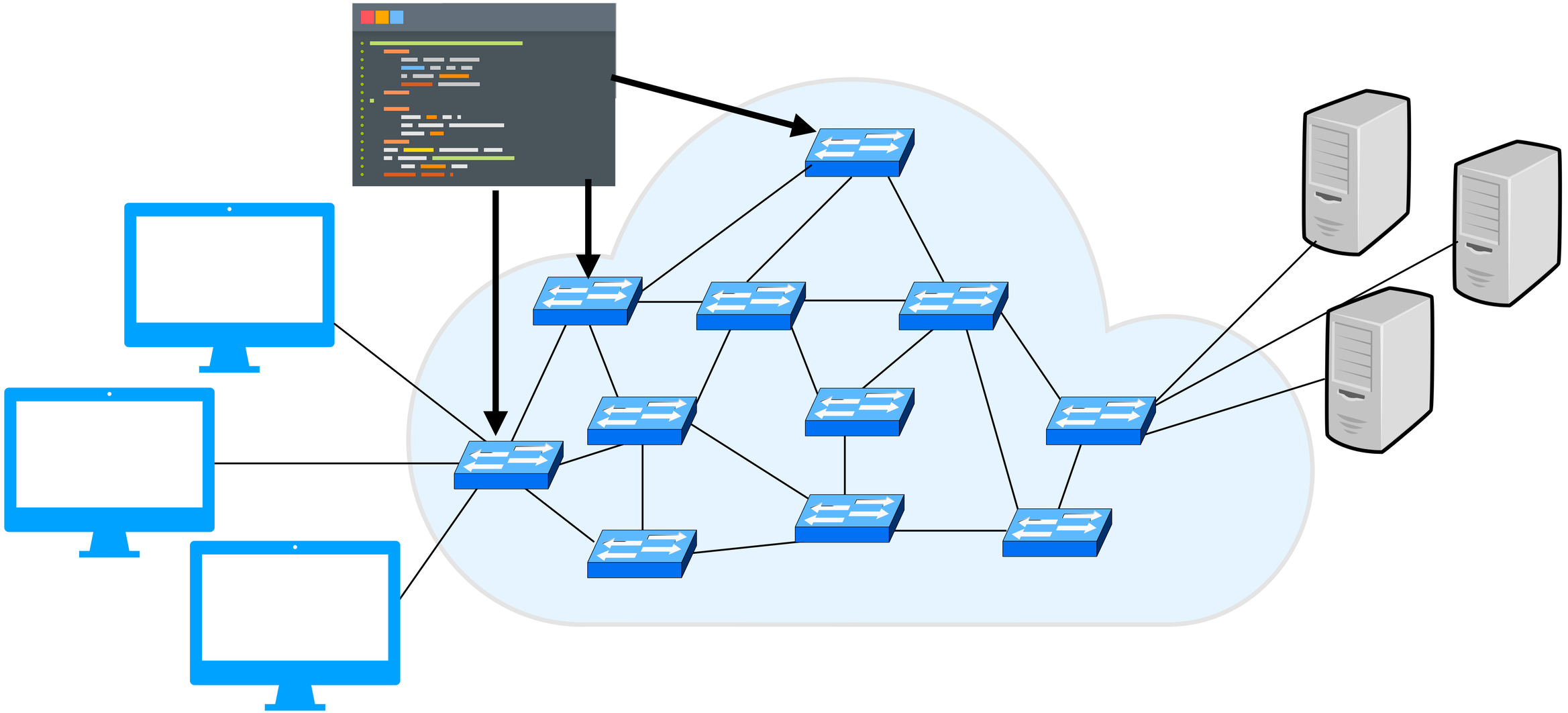
Ready to run in the network!

Future Work: Programming the ***Entire*** Network

Single-Device Programming



Single-Device Programming



Future Work: Programming the ***Entire*** Network

A system that:

(1) Decides **where** to put the program

(2) Finds the optimal resource allocation across **multiple** network devices

Goal: Programmers can easily describe how network-wide applications should perform without grappling with low-level details.

NOT only:

Programmable switches

Linux kernel

SmartNICs

A Broader Device-Independent Approach

Language abstraction for expressiveness

Device-independent language

Optimization for scarce resources

Automatically allocate scarce resources across devices

Compile to domain-specific code

Code that adheres to restrictions

Compile to hardware binary

Ready to run in the network!

Thank you!


Jen Rexford


Dave Walker


Shir Landau Feibish


John Sonchack


Devon Loehr


Mina Tahmasbi Arashloo


Rob Harrison


Maria Apostolaki


Wei Luo


Sharad Agarwal


Rachee Singh


Ryan Beckett


Gerry Wan


Yiming Qiu

Language Expressiveness Under Extreme Scarcity in Programmable Data Planes

Mary Hogan

Backup slides

P4All ILP Constraints

Variables	
Actions	#1 $\{x_{a_i,s} \mid 0 \leq s < S\}$
Registers	#2 $\{m_{r_i,s} \mid 0 \leq s < S\}$
Match-Action Tables	#3 $\{tm_{t_i,s} \mid 0 \leq s < S\}$
Metadata	#4 $\{d_i \mid i \leq U_v\}$
Constraints	
Dependencies	
Same-Stage	#5 $x_{a_i,s} = x_{b_i,s} s < S$
Exclusion	#6 $x_{a_i,s} \leq 1 - x_{b_i,s}$ $s < S$
Precedence	#7 $x_{b_i,y} \leq 1 - x_{a_i,z}$ $y, z < S, y \leq z$
Conditional	#8 $\sum_{0 \leq s < S} x_{a_i,s} = \sum_{0 \leq s < S} x_{b_i,s}$ $0 \leq i \leq U_v$
Resources	
Memory	#9 $\sum_i m_{r_i,s} \cdot w_{r_i} \leq M \quad \forall s < S$
	#10 $m_{r_i,s} \leq x_{a_i,s} \cdot M \quad 0 \leq s < S$
	#11 $m_{r_i,s} \cdot w_0 = m_{0,s} \cdot w_{r_i}$ $\forall s < S, r \geq 1$
TCAM	#12 $\sum_i tm_{t_i,s} \cdot tw_{t_i} \leq T \quad \forall s < S$
Stateful ALUs	#13 $\sum_i H_f(a_i) \cdot x_{a_i,s} \leq F$ $\forall 0 \leq s < S$
Stateless ALUs	#14 $\sum_i H_l(a_i) \cdot x_{a_i,s} \leq L$ $\forall 0 \leq s < S$
PHV	#15 $\sum_i d_i \cdot bits_d \leq P - P_{fixed}$
	#16 $d_i = \sum_{0 \leq s < S} x_{a_i,s}$ if accesses(a,d)
Hash Functions	#17 $\sum_i h_{ha_i,s} \leq N \quad \forall s < S$
Others	
At Most Once	#18 $\sum_{0 \leq s < S} x_{a_i,s} \leq 1$
Inelastic Actions	#19 $\sum_{0 \leq s < S} x_{a_{ne},s} = 1$

The System

P4All
[HotNets'20]
Parasol
[In Submission]

Language abstraction for expressiveness

Did P4All reduce code repetition?

P4All
[NSDI'22]
Parasol
[In Submission]

Optimization with scarce resources

Automatically tailor program to environment

Compile to domain-specific code (P4)

P4 code that fits switch architecture

Programmable switches

Compile to hardware binary

Ready to run in the network!

Repetition Reduction

Application	**P4 LoC**	**P4All LoC**
CMS		
Key-value store		
Key-value store + CMS		
Switch.p4		
IP forwarding + stateful firewall		
Beaucoup		
Precision		
NetChain		
SketchLearn		
Conquest		
Bloom filter		

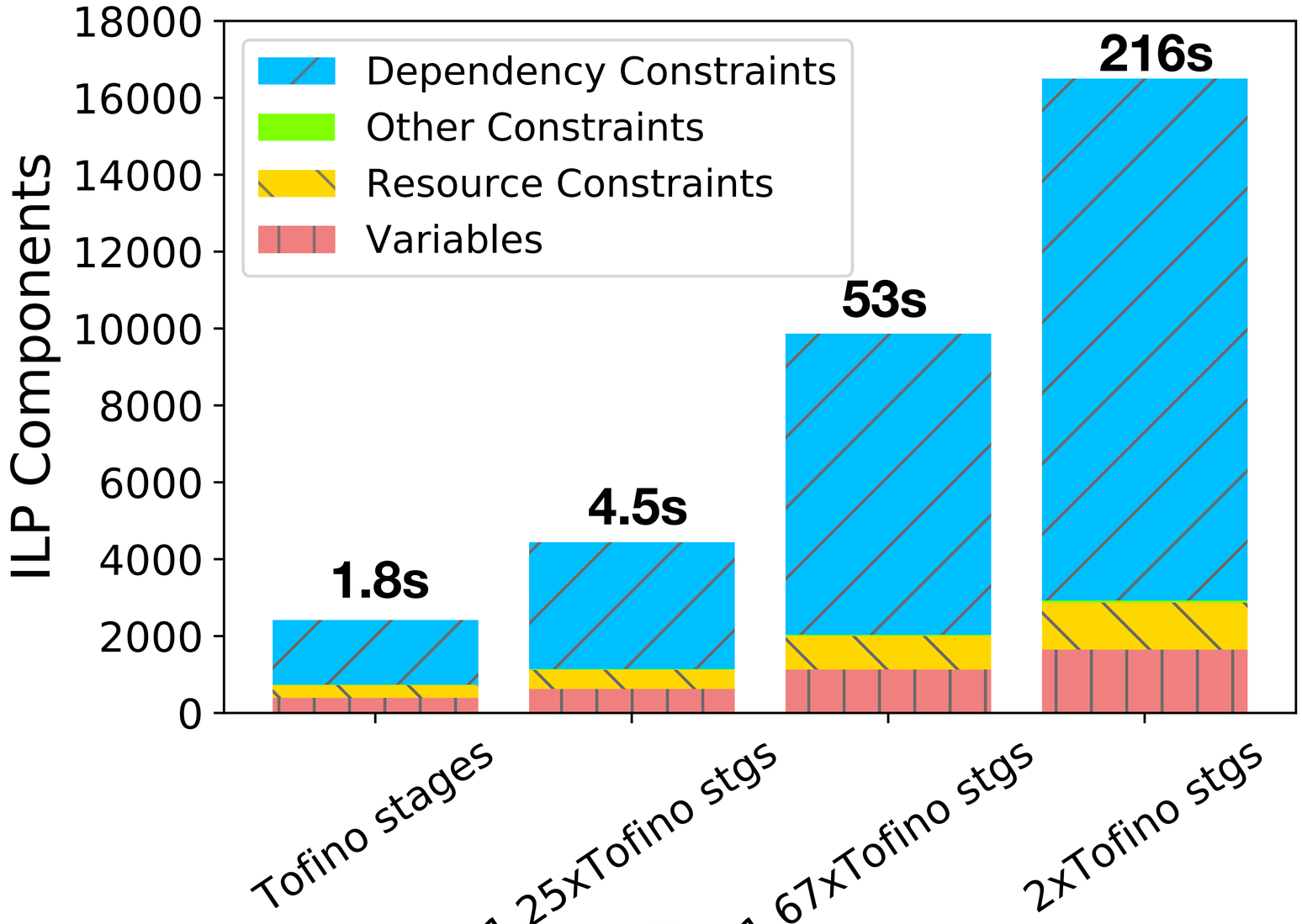
Repetition Reduction

Application	**P4 LoC**	**P4All LoC**
CMS	207	179
Key-value store		
Key-value store + CMS	216	170
Switch.p4		
IP forwarding + stateful firewall		
Beaucoup		
Precision	366	273
NetChain		
SketchLearn		
Conquest		
Bloom filter	179	70

Repetition Reduction

Application	P4 LoC	P4All LoC
CMS	207	179
Key-value store	127	127
Key-value store + CMS	216	170
Switch.p4	804	804
IP forwarding + stateful firewall	282	282
Beaucoup	541	541
Precision	366	273
NetChain	242	242
SketchLearn	445	445
Conquest	869	869
Bloom filter	179	70

ILP Overhead



P4All vs Parasol Language

	P4All	Parasol
Base language	P4	Lucid
Parameters	Symbolic integers	Symbolic values (integers, booleans, etc.)
Objectives	Closed-form objective functions	Python objective functions
Optimization	ILP	Simulation

Compilation Strategies

Strategy	Resources Considered	Average compile time
Full Compilation	All switch resources	3 min

Compilation Strategies

Strategy	Resources Considered	Average compile time
Full Compilation	All switch resources	3 min
Dependency Graph (P4All)	Action dependencies, pipeline stages	51 s

Compilation Strategies

Strategy	Resources Considered	Average compile time
Full Compilation	All switch resources	3 min
Parasol to P4	All resources except PHV	1.5 min
Dependency Graph (P4All)	Action dependencies, stages	51 s

Compilation Strategies

Strategy	Resources Considered	Average compile time
Full Compilation	All switch resources	3 min
Parasol to P4	All resources except PHV	1.5 min
Greedy Layout	Action dependencies, stages, memory, hash units, array accesses, ALUs	51 s
Dependency Graph (P4All)	Action dependencies, stages	51 s

Compilation Times Per App

Application	Full Compilation	Parasol to P4	Greedy Layout	Dataflow Graph
Cache	43 s	4.5 min	3.5 min	3.5 min
Hash Table	23 s	3 s	1 s	1 s
Stateful Firewall	23 s	1.5 min	6 s	6 s
Load Balancer	9 s	1 s	1 s	1 s
Count-Min Sketch	56 s	4 s	1 s	1 s
Precision	1 min	3.5 min	4 min	4 min
Measurement Queries	1.5 min	9 s	7 s	7 s
Congestion Detection	5 min	3 min	2 s	2 s
(Short) RTT Measurement	16 s	49 s	3 s	3 s
(Long) RTT Measurement	24 s	50 s	3 s	3 s

Solutions Per App

Application	Full Compilation	Parasol to P4	Greedy Layout	Dataflow Graph
Cache	1167	1167	1179	1573
Hash Table	60	60	60	65
Stateful Firewall	24	25	25	26
Load Balancer	13	13	13	13
Count-Min Sketch	142	143	143	143
Precision	36	36	36	39
Measurement Queries	5000	5065	5065	10816
Congestion Detection	17	25	25	26
(Short) RTT Measurement	24	24	24	26
(Long) RTT Measurement	13	13	13	13

Search Strategy Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

Do search strategies find the optimal parameter values?

Search Strategy Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

Do search strategies find the optimal parameter values?

Optimal parameters: simulate every set of parameters in reduced search space

Search Strategy Evaluation

Application	Optimal Parameters?
Cache	
Hash Table	✓
Stateful Firewall	✓
Load Balancer	
Count-Min Sketch	✓
Precision	✓
Measurement Queries	✓
Congestion Detection	
(Short) RTT Measurement	✓
(Long) RTT Measurement	

Search Strategy Evaluation

Application	Optimal Parameters?
Cache	
Hash Table	✓
Stateful Firewall	✓
Load Balancer	
Count-Min Sketch	✓
Precision	✓
Measurement Queries	✓
Congestion Detection	
(Short) RTT Measurement	✓
(Long) RTT Measurement	

Search Strategy Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

$$\frac{Opt\_score - Strategy\_score}{Opt\_score}$$

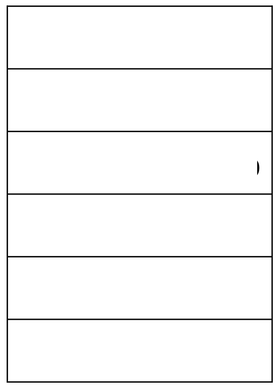
Search Strategy Evaluation

Application
Cache
Hash Table
Stateful Firewall
Load Balancer
Count-Min Sketch
Precision
Measurement Queries
Congestion Detection
(Short) RTT Measurement
(Long) RTT Measurement

$$\frac{Opt\_score - Strategy\_score}{Opt\_score} \quad <= 12\%$$

Parasol vs Hand-Optimized Code

RTT Measurement



RTT = response timestamp
- request timestamp

M : size of data structure

p : probability of adding new request to data structure

Parasol optimized parameters

Max percentile error = 28%

Hand-optimized parameters

Max percentile error = 31%

Unbiased Delay Measurement, *Zheng et al.* [APOCS'22]