Have an awesome idea

Write it in P4

Cannot fit in hardware

Cannot fit in hardware
Why is this so hard?

Programmer
high-level P4 code

Compiler

Switch
Limited resources
Who should fix this?
Who should fix this?

Compiler
Who should fix this?

Knows intended functionality
Ignores hardware specifics
Who should fix this?

Compiler

Knows hardware constraints
Assumes all possible inputs

Knows intended functionality
Ignores hardware specifics
There is a gap between the programmer and the compiler.

- **Programmer** knows intended functionality and ignores hardware specifics.
- **Compiler** knows hardware constraints and assumes all possible inputs.
P²GO bridges the gap between the programmer and the compiler

**P²GO**
- Knows intended functionality
- Ignores hardware specifics

**Compiler**
- Knows hardware constraints
- Assumes all possible inputs
P²GO optimizes the P4 programs for the expected case

- Knows intended functionality
- Ignores hardware specifics

- Compiler
- Knows hardware constraints
  Assumes all possible inputs
P²GO uses the program’s profile to approximate the program’s intended functionality.
P²GO probes the compiler to check that P4’ uses less hardware resources.
$P^2$GO asks the programmer to verify that $P4'$ satisfies the intended functionality.
P²GO: P4 Profile-Guided Optimizations

Patrick Wintermeyer  Maria Apostolaki  Alexander Dietmüller  Laurent Vanbever
**P²GO: P4 Profile-Guided Optimizations**

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary evaluation

Open research questions
P\textsuperscript{2}GO: P4 Profile-Guided Optimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary evaluation

Open research questions
What is the program’s profile?
What is the program’s profile?

...is a description of the program’s behavior during runtime
What is the program’s profile?

...is a description of the program’s behavior during runtime

...contains the control paths that packets of a realistic traffic trace take in the program’s control logic
What is the program’s profile?

...is a description of the program’s behavior during runtime.

...contains the control paths that packets of a realistic traffic trace take in the program’s control logic.
What is the program’s profile?

- Mutually-exclusive actions
- Hot code segment

Program

Profile-based observations
P²GO obtains the profile offline
To obtain the profile P²GO first instruments the P4 program
To obtain the profile P²GO first instruments the P4 program

This packet passed through actions X, Y, Z...
To obtain the profile P²GO first instruments the P4 program
Next P²GO runs it on a traffic trace and collects the output
The profile contains the hit rate of each action and the non-mutually exclusive actions.
OPTIMIZATIONS

OPTIMIZATIONS EVERYWHERE
P²GO uses three profile-guided optimizations to reduce the number of stages occupied by a P4 program.
P²GO uses three profile-guided optimizations to reduce the **number of stages** occupied by a P4 program.
The programmer implements a program in P4

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
The programmer implements a program in P4

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
The programmer implements a program in P4

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
The compiler maps the program to hardware using five stages

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3

5 stages
SYNFWD needs to precede the execution of MAGIC_1

If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3

To execute MAGIC_1*
we need to have results of SYNFWD.
MAGIC_2 uses more memory than is available on a single stage

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3

Too many M&A rules.

Compiler

5 stages
P²GO uses the program’s profile and the compiler’s output to reduce the number of stages used by the example program.
P²GO uses the profile to reduce the number of stages, while not changing the program’s semantic.

- Increase pipeline concurrency
- Reduce resource waste
- Improve hardware–software split

remove fake dependencies
P²GO extracts the program’s dependencies from the compiler

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
P²GO compares the profile with the dependencies of the static analysis.

If `tcp.flags` == SYN:
- SYNFWD
If `tcp.len` > 10:
- MAGIC_1
- MAGIC_2
- MAGIC_3

No packet matched both SYNFWD & MAGIC_1.
P²GO automatically generates a new program with the dependency resolved.

```python
If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
```

P²GO

Compiler
P²GO verifies that the change will reduce the hardware allocation.

If tcp.flags==SYN:
  SYNFWD
Else If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3
P²GO asks the programmer to accept the modification, explaining the profile-based observation that triggered it.

If you are sure that SYNFWD & MAGIC_1 are never applied to the same packet you can gain a stage.

If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
The programmer examines and accepts the modification

If you are sure that `SYNFWD & MAGIC_1` are never applied to the same packet you can gain a stage.

Ah yes, SYN packets have zero payload

If `tcp.flags==SYN:`
SYNFWD
Else If `tcp.len >10`:
MAGIC_1
MAGIC_2
MAGIC_3
P²GO uses the profile to reduce the number of stages, while not changing the program’s semantic.

- Increase pipeline concurrency
- Remove fake dependencies
- Reduce resource waste
- Reduce memory usage
- Improve hardware-software split
P²GO fetched the most seldom used tables from the profile

If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3

MAGIC_2 is seldom used
MAGIC_3 is seldom used
P²GO generates programs with reduced memory and resubmits them.

- **MAGIC_1**
- **MAGIC_2** - Seldom used
- **MAGIC_3** - Seldom used

If `tcp.flags==SYN`:
- SYN_FWD

If `tcp.len > 10`:
- MAGIC_1
- MAGIC_2
- MAGIC_3
  - MAGIC_2 is seldom used
  - MAGIC_3 is seldom used

P²GO → Compiler

- **P4'** Smaller Magic_2 → 1 stage less
- **P4''** Smaller Magic_3 → 0 stage less
P²GO repeats the process to find the minimum change to save a stage

If tcp.flags==SYN:
SYNFWD
If tcp.len >10:
MAGIC_1
MAGIC_2
MAGIC_3

MAGIC_2 is seldom used
MAGIC_3 is seldom used

-5%  -1%  -2%
2% less mem saves 1 stage
P²GO asks the programmer whether he would accept the reduction

With 2% less memory for MAGIC_2 you can save a stage.

If tcp.flags==SYN:
SYNFWD
If tcp.len >10:
MAGIC_1
MAGIC_2
MAGIC_3

MAGIC_2 is seldom used
MAGIC_3 is seldom used

2% less mem saves 1 stage

Compiler
The programmers considers and accepts the change

Ah sure that was a rough estimate anyway.

With 2% less memory for MAGIC_2 you can save a stage.

If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3

MAGIC_2 is seldom used
MAGIC_3 is seldom used

P²GO

2% less mem saves 1 stage

Compiler

-5%
-1%
-2%
P²GO uses the profile to reduce the number of stages, while not changing the program’s semantic.

- Increase pipeline concurrency
- Reduce resource waste
- Improve hardware–software split
- Remove fake dependencies
- Reduce memory usage
- Migrate code segments to software
P²GO fetches the least-used self-contained segment

```
If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3
```

MAGIC_3 is self-contained & seldom used.
P²GO generates a program without MAGIC_3 that sends the corresponding packets to the controller.

If tcp.flags==SYN:
  SYN_FWD
If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3

MAGIC_2 is self-contained & seldom used.
P²GO generates a program without MAGIC_3 that sends the corresponding packets to the controller.

MAGIC_3 is seldom used and self-contained, by moving it to software you can save a stage.

If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
    MAGIC_1
    MAGIC_2
    MAGIC_3

MAGIC_3 is self-contained & seldom used.
P^2GO generates a program without MAGIC_3 that sends the corresponding packets to the controller.

MAGIC_3 is seldom used and self-contained. By moving it to software you can save a stage.

Ah no! MAGIC_3 will stay in the data plane. The traffic trace used for profiling happened to not contain such traffic.

If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3

MAGIC_3 is self-contained & seldom used.
P²GO uses the profile to reduce the number of stages, while not changing the program’s semantic.

- Increase pipeline concurrency
- Reduce resource waste
- Improve hardware–software split

- remove fake dependencies
- reduce memory usage
- migrate code segments to software
P2GO: P4 Profile-Guided Optimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware–software split

Preliminary evaluation

Open research questions
P²GO working alongside the Tofino compiler reduces the pipeline length of realistic examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Used Optimization</th>
<th># Stages Before</th>
<th># Stages After</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAT &amp; GRE</td>
<td>Removing Dependencies</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Sourceguard</td>
<td>Reducing Memory</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Failure Detection</td>
<td>Offloading Code</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
P2GO: P4 Profile-Guided Optimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary Evaluation

Open research questions
Open research questions

Representative traffic trace

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions
Open research questions

Representative traffic trace

How can we find a representative traffic trace? Would the problem be solved with online profiling?

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions
Open research questions

Representative traffic trace

**Semantic equivalence**

Can we ensure that the optimized program is semantically equivalent without involving the programmer?

Mis-speculation

Optimize across multiple dimensions
Open research questions

Representative traffic trace

Semantic equivalence

**Mis-speculation**

Optimize across multiple dimensions

How to detect and mitigate inaccuracies of the profile or of the programmer?
Open research questions

Representative traffic trace

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

What if a program does not compile due to other resources? How to optimize the program across multiple dimensions?
P²GO \textbf{P4 Profile-Guided Optimizations}

1. pcap
2. Profile
3. #stages
4. P²GO
5. Profile-based Observation

Compiler