CacheP4: A Behavior-level Caching Mechanism for P4

Zijun Ma, Jun Bi, Cheng Zhang, Yu Zhou, Abdul Basit Dogar*
Tsinghua University

CCS CONCEPTS

• **Networks → Programmable networks**;

KEYWORDS

P4, data plane, caching mechanism, match action table

ACM Reference format:
Zijun Ma, Jun Bi, Cheng Zhang, Yu Zhou, Abdul Basit Dogar. 2017. CacheP4: A Behavior-level Caching Mechanism for P4. In *Proceedings of SIGCOMM Posters and Demos '17, Los Angeles, CA, USA, August 22–24, 2017,* 3 pages.
DOI: 10.1145/3123878.3132003

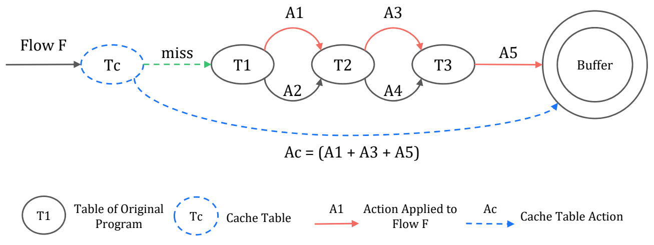


Figure 1: Behavior-level cache.

1 INTRODUCTION

The P4 programming language [1] offers the flexibility of defining data plane behaviors by applying a pipeline of match+action tables (MATs) on packets. *For a P4 program without stateful memories (e.g., counters, meters, and registers), packets in an identical flow will be processed by the same behavior (i.e., the same sequence of compound actions).* For instance, in Figure 1, packets in flow F will be processed by compound actions A_1, A_3 and A_5 in an ingress pipeline. Since the first packet in flow F has already identified the corresponding switch behavior, it is a waste for all the MATs to perform lookups on the header fields of subsequent packets in flow F.

Inspired by the idea of caching, we put a MAT T_c in the front of the pipeline. T_c could recognize a packet in flow F by performing a lookup on packet header fields. Consequently, a compound action A_c consisting of A_1, A_3 and A_5 is applied to the packet. In such a case, T_c serves as a table for behavior-level cache – it contains the feature of flow F as well as the corresponding behavior. To the best of our knowledge, up to now, such behavior-level cache has not been introduced to P4 targets (e.g., FPGA and BMv2 [2]), which leaves potential to explore for faster packet processing.

Extending the idea of the behavior-level cache, we propose CacheP4, a caching mechanism for P4. In CacheP4, (1) a cache MAT determined by language elements (e.g., parser definitions, table definitions and control flows) of the original P4 program is inserted into the front of ingress/egress pipeline by a preprocessor before compilation, and (2) the cache MAT is populated during the runtime according to the control plane to accelerate the forwarding of selected flows. Since CacheP4 provides the behavior-level cache by modifying the original P4 program regardless of target specification, it is likely to be a general cache solution on various P4 targets.

2 DESIGN OF CACHEP4

Mechanism overview. As is depicted in Figure 2, (1) during preprocessing stage, the structure of a cache MAT is determined by analyzing language elements in the original P4 program. (2) The P4 program containing cache MATs is then compiled and configured onto a particular P4 target. (3) During the runtime, the control application is notified of selected flows that need to be cached by the P4 target. The notifier could be network monitors or network operators. (4) After receiving the notification, the control application computes cache MAT entries based on the desired packet header and current table entries in the P4 target. (5) Cache MAT is populated by the control application to accelerate the forwarding of selected flows.

Definitions and preliminaries. To concisely describe the algorithm for determining and populating cache MATs, we provide the following definitions and preliminaries:

- P4 variable: A field of metadata or packet header defined in a P4 program.
- P4 factor: A field of standard metadata or packet header defined in a P4 program. It is a special case of P4 variable.
- P4 operation: A language element in a P4 program that reads and/or writes a variable. In P4 parser, a P4 operation could be a *set_metadata* statement or a *select* statement. In P4 pipeline, a P4 operation could be a MAT *reads*, an *if* statement or a primitive action.
- P4 predication: A P4 conditional operation, which could be a *select* statement, an *if* statement or a MAT *reads*.
- P4 path: A sequence of P4 operations by which a packet is likely to be processed. A P4 program usually has multiple P4 paths.
- Packet Trace (PTR): A sequence of P4 MAT entries that a packet is likely to hit during the runtime. A P4 path might correspond to multiple PTRs.

*Zijun Ma, Jun Bi, Cheng Zhang, Yu Zhou and Abdul Basit Dogar are with Institute for Network Sciences and Cyberspace, Tsinghua University. Zijun Ma is also with School of Software, Tsinghua University. Jun Bi, Yu Zhou, Cheng Zhang and Abdul Basit Dogar are also with Department of Computer Science, Tsinghua University, and Tsinghua National Laboratory for Information Science and Technology (TNList). This work is supported by National Key R&D Program of China (2017YFB0801701) and the National Science Foundation of China (No.61472213). Jun Bi is the corresponding author.

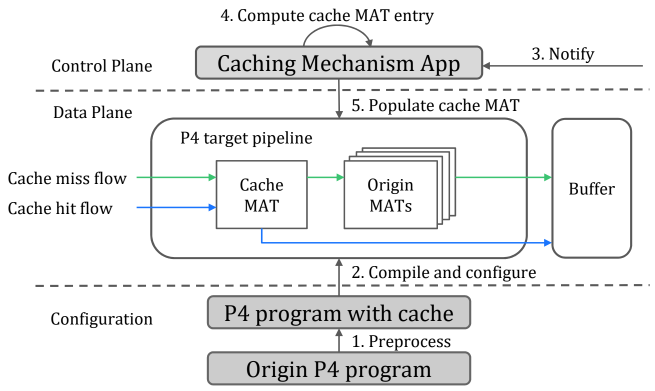


Figure 2: Mechanism overview of CacheP4.

- Match Type Power: The expressing ability of a match type. We conclude *ternary* > *lpm* > *range* > *exact* in terms of match type power.
- Value Impact: Expressed as $I = (P, F, V, R, T)$. I is a value impact; P is a P4 path; F is a P4 factor; V is a P4 variable; R is a P4 predication and T is a match type. This indicates that in path P, the initial value of F has an impact on the value of V read by predication R. If R belongs to an ingress/egress pipeline, I is considered as an ingress/egress value impact. T equals to R's match type or equivalently expresses R's relation operator (e.g., *exact* match for == and *range* match for >=). We consider T as a candidate match type of F.

Determining cache MAT structure. Take a cache MAT T_c of the ingress pipeline for example. Any P4 factor that appears in a value impact of the ingress pipeline should be a match field of T_c. For a match field, we choose the most powerful candidate match type as its match type in T_c. An action of T_c should be a combination of compound actions in a P4 path.

Populating cache MAT. A flow to be cached is described as $E = \{(h.name, h.value) \mid h\ is\ a\ packet\ header\ field\}$ in the notification. After the control application receives the notification, it would compute all PTRs based on the flow description and current table entries in the P4 target. Note that an E might correspond to multiple PTRs, we extend an E to several E's by adding new packet header fields to guarantee that each E' corresponds to one PTR only. For a cache MAT entry T_e, a match field $f_c.value$ is generated based on an E'. To be specific, if $f_c \in E'$, then $f_c.value$ is set according to E'. Otherwise, $f_c.value$ is set to be the "widest" value (e.g., wildcard for *ternary* match and *lpm* match). The action name and action parameters of T_e are determined by the corresponding PTR.

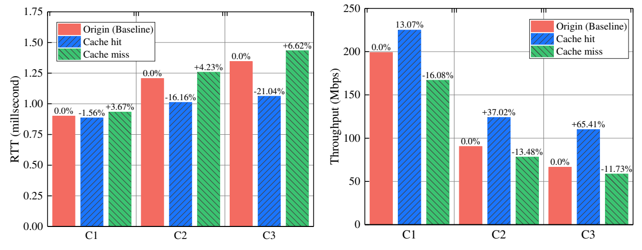
3 EVALUATION OF CACHEP4

We use BMv2 to evaluate the performance of CacheP4 with three test cases [3] which are shown in Table 1. Programs of these cases are based on the P4 *switch* program [4]. For each case, we measure the performance of three different configurations: (1) original P4 program without cache table, (2) P4 program with cache table whose entries lead to a cache hit and (3) P4 program with cache table whose entries lead to a cache miss.

As can be seen from Figure 3(a) and 3(b), with the P4 program becoming more involved, (1) performance improvement under cache hit condition is higher since most MAT match and MAT transfer are avoided. (2) Performance penalty under cache miss condition is relatively lower since delay caused by the cache MAT match takes a smaller proportion of the total packet processing time.

Case	P4 program
C1	Router
C2	NAT + Router
C3	IP Source Guard + ACL + NAT + Router

Table 1: Test cases for CacheP4.



(a) Round-Trip Time. (b) Throughput.

Figure 3: Performance evaluation for CacheP4.

All the experiments above are ideally based on one single packet flow traversing a P4 target. However, multiple packet flows would simultaneously traverse a P4 target in a production environment. Therefore, the strategy to decide which flows should be cached to make the best use of the cache MAT is required. Here we put forward several pertinent factors: (1) pre-estimated number of packets within a flow, (2) packet size within a flow, (3) number of MATs a flow traverses (more MATs mean the gains obtained from caching are bigger), (4) target resource constraint leading to limited cache MAT entries and (5) policy intent.

4 FUTURE WORK

CacheP4 shows great potential for improving packet forwarding speed and throughput of P4 targets. However, more work is required for the design refinement as well as the implementation.

Automatic cache maintenance. We plan to implement a preprocessor for P4 programs to generate cache MAT structure automatically. Note that during the runtime (1) a new flow could be handled by the cache, (2) a cached flow could physically disappear and (3) the original MAT entries could be updated. Therefore, strategies for automatically repopulating cache MAT under such dynamic circumstances call for careful design and implementation.

Customized flow selection. Instead of proposing a particular standard on selecting cached flow, we plan to design a set of interfaces for network operators to designate the desired flow to cache and notify the control application based on their considerations. These interfaces include information such as (1) the flow description, (2) condition to start caching the flow and (3) duration for caching the flow.

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[2] Barefoot Networks. P4 behavioral model. Website. https://github.com/p4lang/behavioral-model.

[3] Cachep4 test cases. Website. https://github.com/mzj14/CacheP4-Test.

[4] P4 Language Consortium. P4 switch. Website. https://github.com/p4lang/switch.