Multi-Stage Flow Table Caching: From Theory to Algorithm

Ying Wan[1], Haoyu Song[2], Tian Pan[3], Bin Liu[4], Yu Jia[1], Ling Qian[1]

[1]China Mobile (Suzhou) Software Technology Co., Ltd, China
[2]Futurewei Technologies, USA
[3]Beijing University of Posts and Telecommunications, China
[4]Tsinghua University, China

Email: wanying-cx@cmss.chinamobile.com

Outline

- Background and related work
- Theoretical Analysis of OMFC
- Approximation algorithm for OMFC
- Performance evaluation
- Conclusion

Background —— flow table in SDN

- **SDN decouples the data plane and the control plane**
  - control plane: generate flow table rules to guide how to handle packets
  - data plane: process packet based on the rules in flow table installed by the control plane
- **Flow table in SDN switches**
  - support multiple matching patterns
    - (e.g., EM, LPM, RM)
  - manipulate packets flexibly
    - (e.g., drop/forward/modify)
  - cannot be fully implemented in the limited on-chip hardware resource
    - (e.g., SRAM and TCAM)

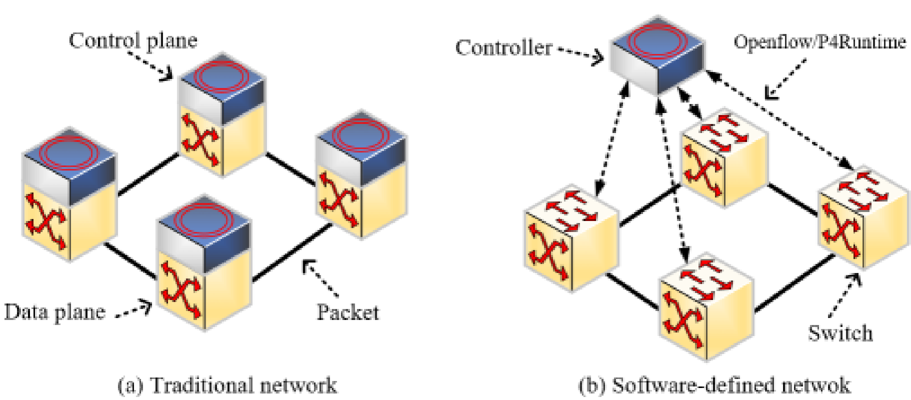


Fig.1 The architecture of different networks.

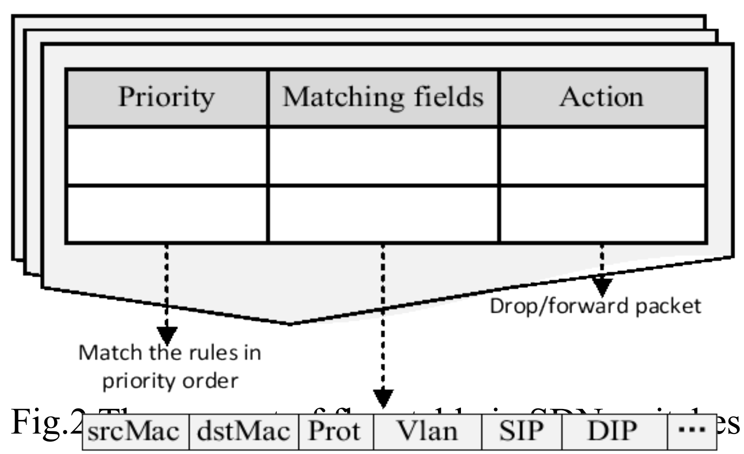


Fig.2

Background —— flow table caching

- **Hardware-based flow table**
  - expensive, small capacity, and incomparable lookup speed
- **Software-based flow table**
  - cheap, large capacity, and low lookup speed
- **hybrid flow table**
  - Using hardware-based flow table as cache
    - Temporal and spatial locality of network traffic (1%flow-->70% traffic --> 0.13 rules)
    - stores only the popular rules that matching a large amount of packets
  - Only missed packets need to be processed by the software flow table.
- **exiting works**
  - pick the caching rules among the flow table to achieve the highest cache hit-rate.

	Software Switch	Hardware Switch
Cost	Cheap	Expensive
Rule capacity	Low (~2K-10K)	High
Rule insertion	High	Low (<50/s)

Fig.3 The comparison of hardware Switch and software Switch

Background —— multi-stage flow table

- **A single flow table may be decomposed into a multi-stage flow table.**
  - These multiple decoupled flow tables form a logical table chain
  - The match in the first i stages leads to the search in the (i+1)-stage flow table
  - reducing flow table size ｜ simplify flow table update | adopt to hardware limitation
- **multi-stage flow table caching**
  - Only when all the stages are matched, a rule is matched
  - high cache hit-rate at each stage does not imply high cache hit-rate of the overall scheme
  - a caching entry can be shared by multiple rules

VPC	VM IP	VPC	NH
VNI=2	10.0.3.1	VNI=5	192.168.1.2
VNI=2	10.0.3.2	VNI=5	192.168.1.7
VNI=4	10.0.3.1	VNI=5	192.168.1.2
VNI=4	10.0.3.2	VNI=5	192.168.1.7
VNI=6	10.0.3.1	VNI=5	192.168.1.2
VNI=6	10.0.3.2	VNI=5	192.168.1.7

(a) 1-stage flow table

VPC	VM IP	VPC
VNI=2	10.0.3.0/24	VNI=5
VNI=4	10.0.3.0/24	VNI=5
VNI=6	10.0.3.0/24	VNI=5

VPC	VM IP	NH
VNI=5	10.0.3.1	192.168.1.2
VNI=5	10.0.3.2	192.168.1.7

(b) 2-stage flow table

Fig.3 The rule relation graph

Fig.4 use the rule graph to guide rule insertion

Theoretical Analysis of OMFC

- Optimal multi-stage flow table caching (OMFC)
  - each rule is the combination of k entries of the k-stage flow tables
    - an entry in each stage can be shared by multiple rules.
    - an entry with a larger counter does not mean a higher priority to cache
  - given the counter of each entry and total hardware resource
  - decide the capacity and content of k-stage flow tables to achieve the highest cache hit-rate
- Complexity of OMFC
  - prove the NP-hardness of OMFC for the first time
  - reduce from the densest k-subgraph problem on the bipartite graph (DkS)
  - OMFC cannot be solved in polynomial time

Rule	Specification			Counter
	F1	F2	F3	
r_1	e_1^1	e_1^2	e_2^3	7
r_2	e_2^1	e_3^2	e_1^3	7
r_3	e_2^1	e_3^2	e_3^3	6
r_4	e_2^1	e_2^2	e_3^3	4
r_5	e_2^1	e_4^2	e_1^3	3
r_6	e_1^1	e_3^2	e_1^3	2

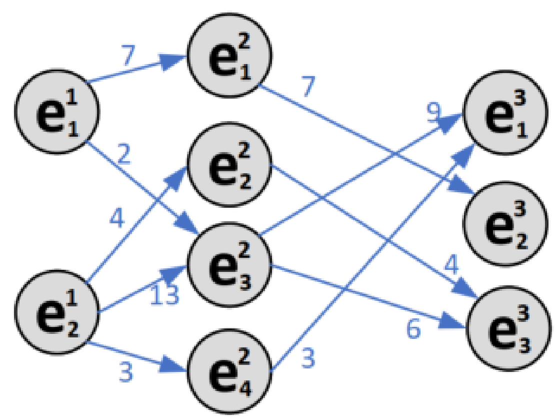
Approximation algorithm for OMFC

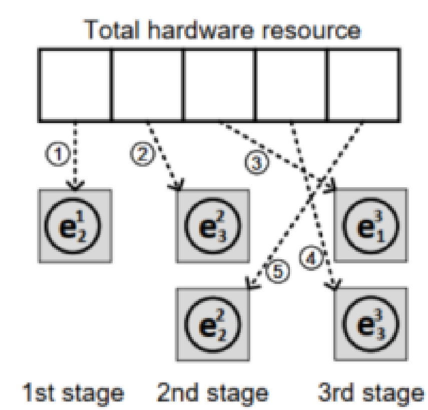
- Entry sharing Graph (ESG)
  - abstract each entry into a vertex, add k-1 directed edges between the vertices of each rule
  - the counter of entry equals to the sum of the counter of the rules passing through it.
  - given the popularity of each entry and total hardware resource
- Cache Profit Calculation
  - focus on the counter of the r itself r.cnt and its k entries r[0].cnt, ..., r[k-1].cnt.
  - the higher the value of r[i].cnt - r.cnt, the higher the concomitant profit of caching r[i].

$$r.pf = r.cnt + \sum_{i=0}^{k-1} \frac{r[i].cnt - r.cnt}{\prod_{j=0}^{i-1} size(F_j) \prod_{j=i+1}^{k-1} size(F_j)}$$

- Greedy Entry Selection
  - select the most profitable rule that the hardware accommodates among the uncached rules.

Rule	Specification			Counter	Profit
	F1	F2	F3		
r_1	e_1^1	e_1^2	e_2^3	7	$7+\frac{2}{8}$
r_2	e_2^1	e_3^2	e_1^3	7	$7+\frac{13}{12}+\frac{8}{6}+\frac{5}{8}$
r_3	e_2^1	e_3^2	e_3^3	6	$6+\frac{14}{12}+\frac{9}{6}+\frac{4}{8}$
r_4	e_2^1	e_2^2	e_3^3	4	$4+\frac{16}{12}+\frac{6}{8}$
r_5	e_2^1	e_4^2	e_1^3	3	$3+\frac{17}{12}+\frac{9}{8}$
r_6	e_1^1	e_3^2	e_1^3	2	$2+\frac{7}{12}+\frac{13}{6}+\frac{10}{8}$





Performance evaluation

- Experiment setup
  - Compare objects: *CacheFlow*[1], PipeCache[2]
  - Testbed: a server with the Ubuntu 16.04-LTS operating system.
  - Dataset： flow tables and traffics generated by ClassBench and classbench-ng.
  - Metric: cache hit-ratio.

Table.I Comparison of interrupt time

Type	Source	Rule #	Stage #	Packet #
ACL	ClassBench-ng	~40K	$3 \sim 6$	7.6×10^5
Firewall	ClassBench-ng	~40K	$3 \sim 6$	5.0×10^5
IP Chain	ClassBench-ng	~40K	$3 \sim 6$	2.1×10^6
Openflow	ClassBench-ng	~40K	$3 \sim 6$	1.2×10^6

[1] “RuleTris: minimizing rule update latency for TCAM-based SDN switches”, ICDCS 2016, best paper
[2] “Partial order theory for fast TCAM updates”, IEEE TON, 2017

Cache hit-rate

- different flow tables
- different cache sizes

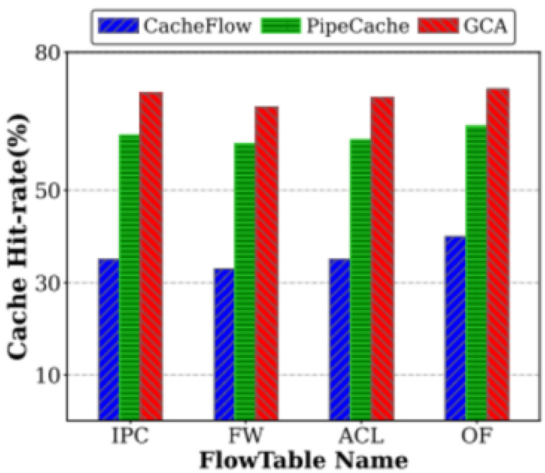
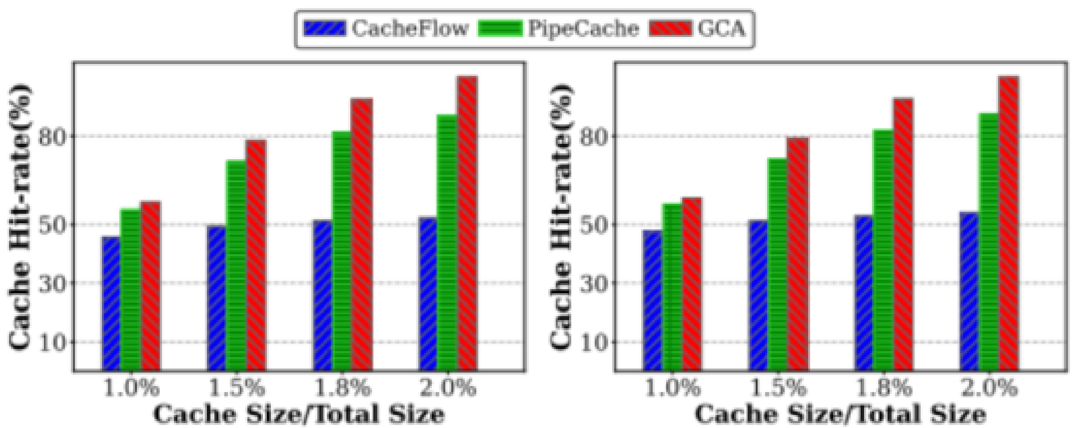


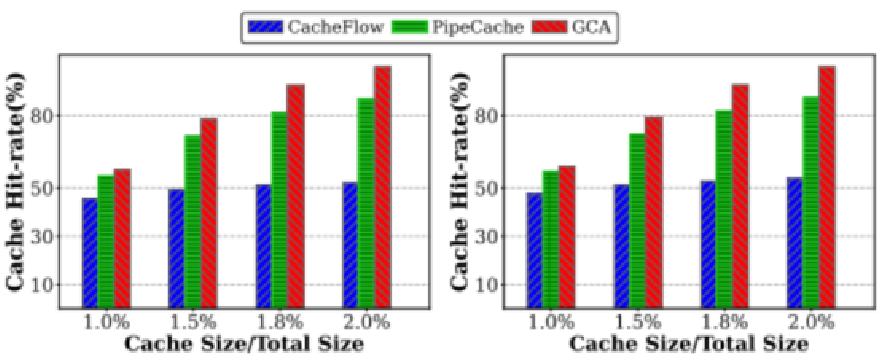
Fig.8 Comparison of compute time



[1] “RuleTris: minimizing rule update latency for TCAM-based SDN switches”, ICDCS 2016, best paper
[2] “Partial order theory for fast TCAM updates”, IEEE TON, 2017

Cache hit-rate

- different entry sharing ratios
- different flow table stages



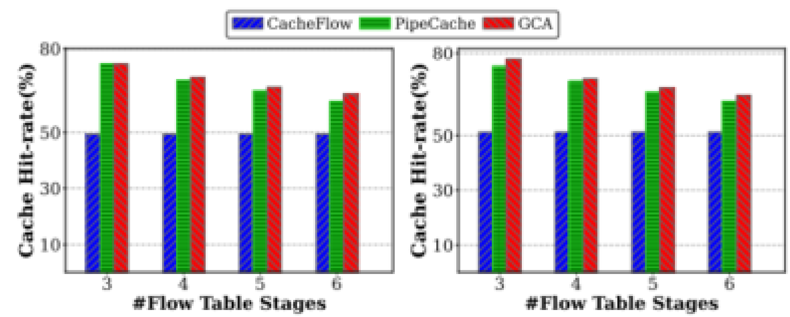


Fig.8 Comparison of compute time

Conclusion

- abstract and models the problem of OMFC
- prove the NP-hardness of OMFC for the first time
- propose algorithm GCA to achieve higher cache hit-rate

Thank You!

Q & A