CutSplit: A Decision-Tree Combining Cutting and Splitting for Scalable Packet Classification

Wenjun Li†, Xianfeng Li†, Hui Li† and Gaogang Xie‡

†School of Electronic and Computer Engineering, Peking University
‡Institute of Computing Technology, Chinese Academy of Sciences

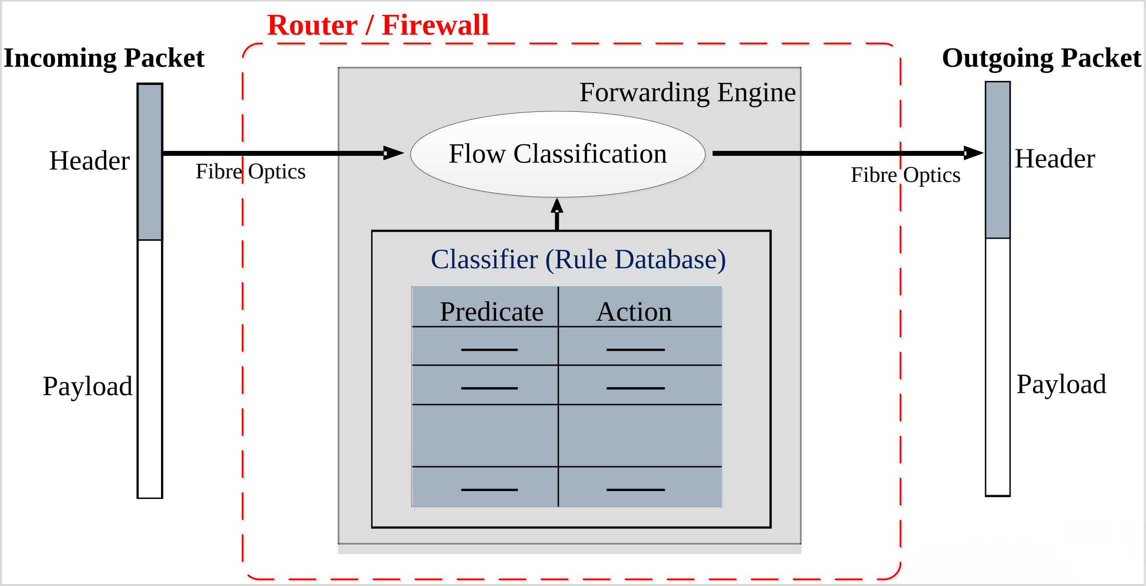
IEEE INFOCOM
Honolulu, HI, April 15-19, 2018

Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion

Multi-field Packet Classification

- **Key for policy enforcement in packet forwarding**
  - Firewall, QoS, OpenFlow, P4, etc.



#	An example OpenFlow 1.0 classifier/flow table (12-tuple)						Action
r_1	**Ingress Port**	**Ether src**	**Ether dst**	**Ether type**	**VLAN id**	**VLAN priority**	$Action_1$
	3	*	*	*2048*	*	*	
	IP src	**IP dst**	**IP proto**	**IP ToS bits**	**TCP/UDP Src Port**	**TCP/UDP Dst Port**	
	15.25.70.8/30	*18.15.125.3/28*	*0x11/0xff*	*1*	*1024 : 65535*	*80*	

…

Previous Works & Key Metrics

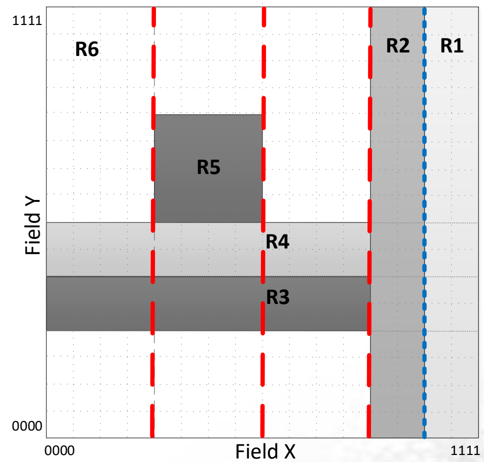
- **Taxonomy of previous packet classifications**
  - **Algorithmic**: Desired but speed/memory inefficient
  - **Architectural**: Fast but expensive, power hungry, poor scalability and suffer from range expansion
- **Key metrics of scalable packet classification**
  - Low memory consumption
  - Low memory accesses
  - Bounded worst-case performance
  - Low pre-processing time
  - Low incremental update time
- **Our proposed algorithm: CutSplit**
  - A decision-tree based algorithmic approach

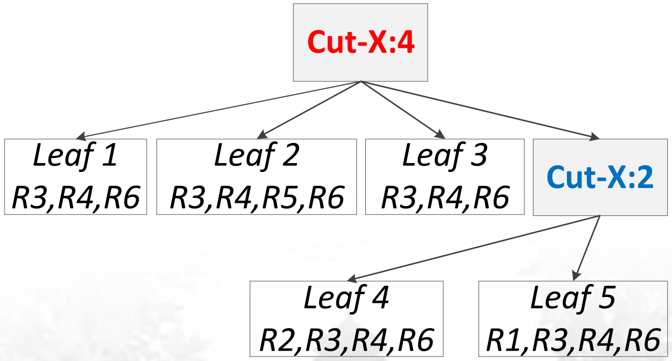
A Little Review on Decision-trees

Decision-tree construction in packet classification

- 1. Rule table matching ↔ Point location in geometric space
- 2. Partition the searching space into sub-spaces recursively
  - Root node: Whole searching space containing all rules
  - Internal node: #rule covered by sub-space > a predefined number of rules
  - Leaf node: #rule covered by sub-space <= a predefined number of rules

Rule #	Field X	Field Y	*Action*
R1	111*	*	*A1*
R2	110*	*	*A2*
R3	*	010*	*A3*
R4	*	011*	*A4*
R5	01**	10**	*A5*
R6	*	*	*A6*





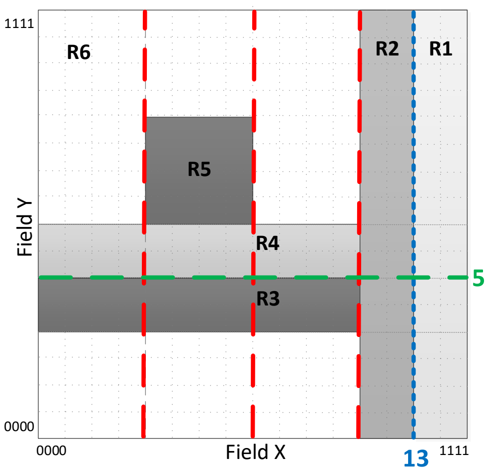
HiCuts-4

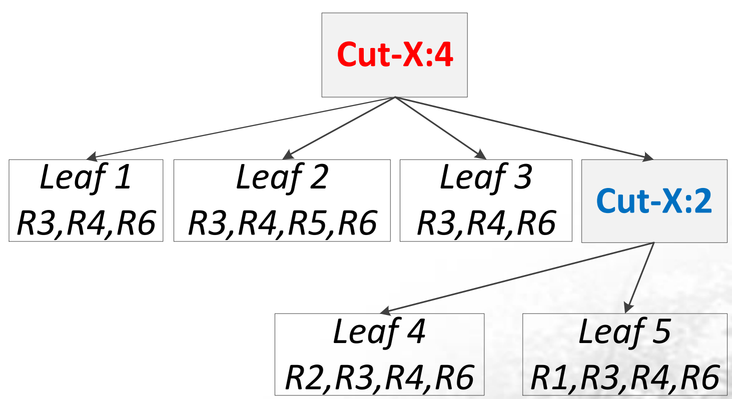
Two major threads of building decision-trees

- **Equal-sized cutting & Equal-dense splitting**

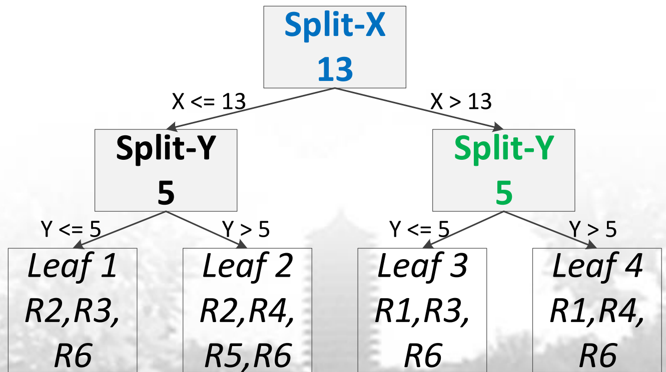
Two Major Threads in Decision-trees

- **Equal-sized cutting based decision-trees**
  - Separate the searching space into many equal-sized sub-spaces
  - e.g., HiCuts, HyperCuts, EffiCuts, HybridCuts, etc.
- **Equal-dense splitting based decision-trees**
  - Unequal-sized sub-spaces containing nearly equal number of rules
  - e.g., HyperSplit, ParaSplit, SmartSplit, etc.





HiCuts-4



HyperSplit-4

Why Yet Another Decision-tree?

A well established problem

without

Well established solutions

Scalability	HyperSplit	EffiCuts	HybridCuts	SmartSplit
Memory consumption	X	√	√	√
Memory accesses	X	X	√	√
Worst-case performance	√	X	X	X
Pre-processing time	X	X	X	X
Incremental update	X	X	X	X

Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion

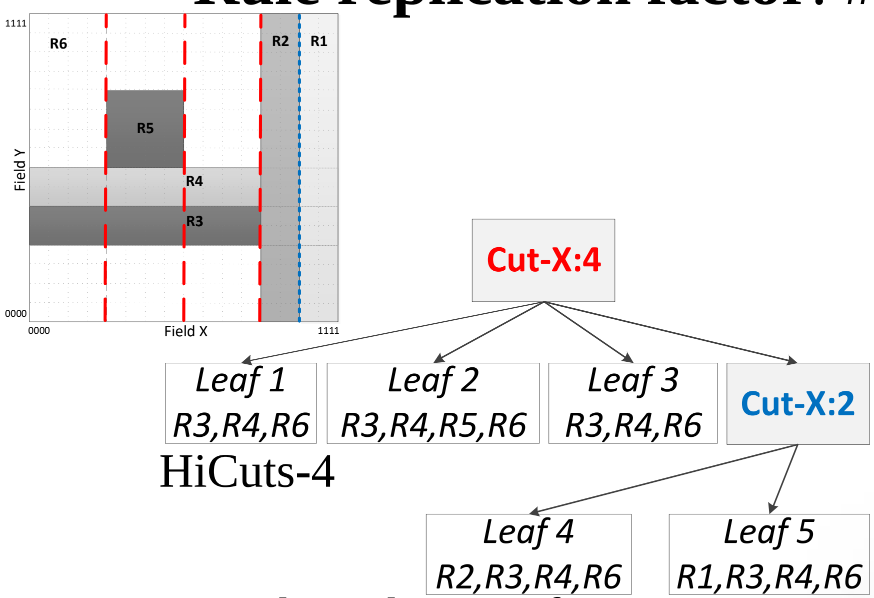
Review & Analysis on Challenges

Rule replication: Main trouble-maker for decision-trees

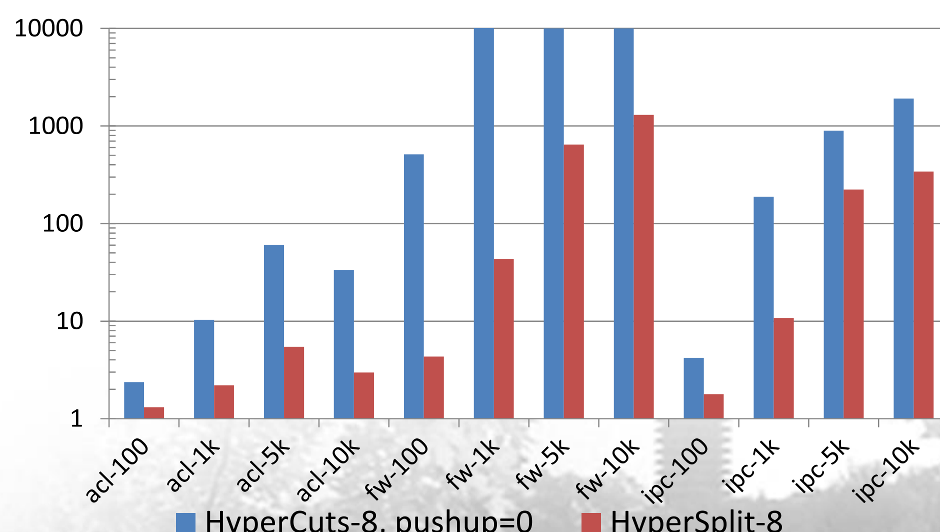
- In case a rule spans multiple sub-spaces/nodes in decision-tree, **rule replication** happens, which is an undesirable case.

More insights on rule replication

- **Rule replication factor**: #stored rules / rule set size



Rule replication factor:
(3+4+3+4+4)/6=3

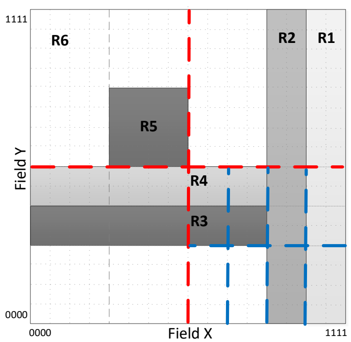


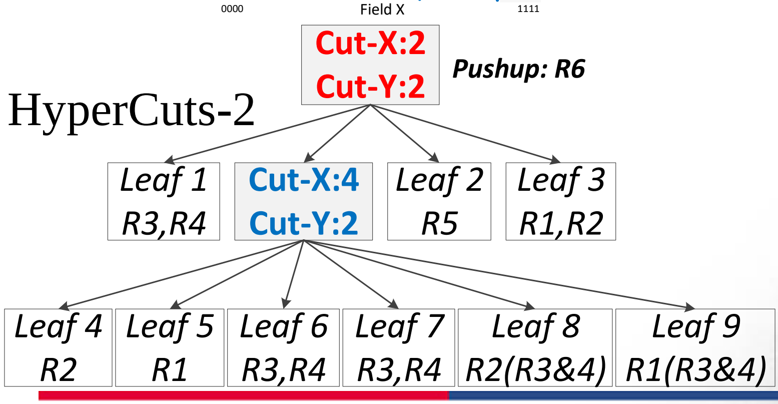
Evaluation of *rule replication factor* for different rule sets

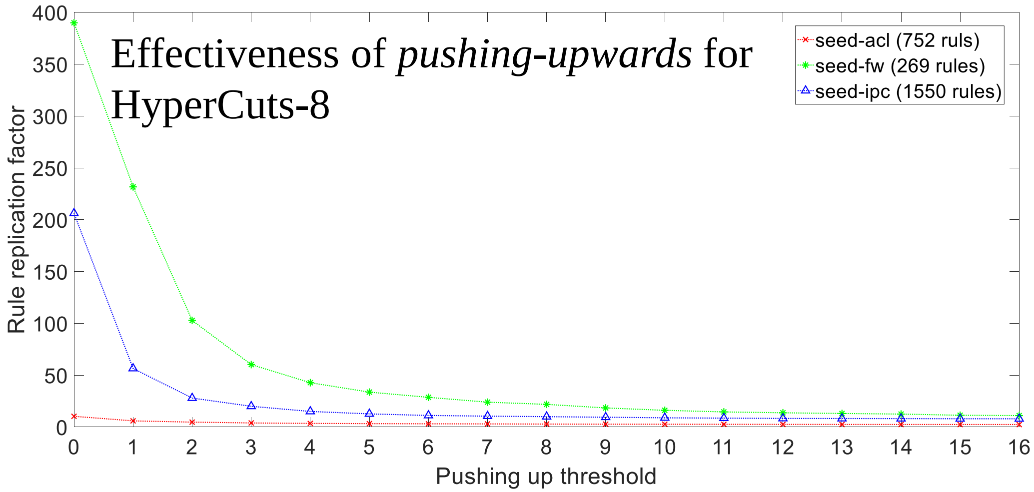
Review & Analysis on Prior Art

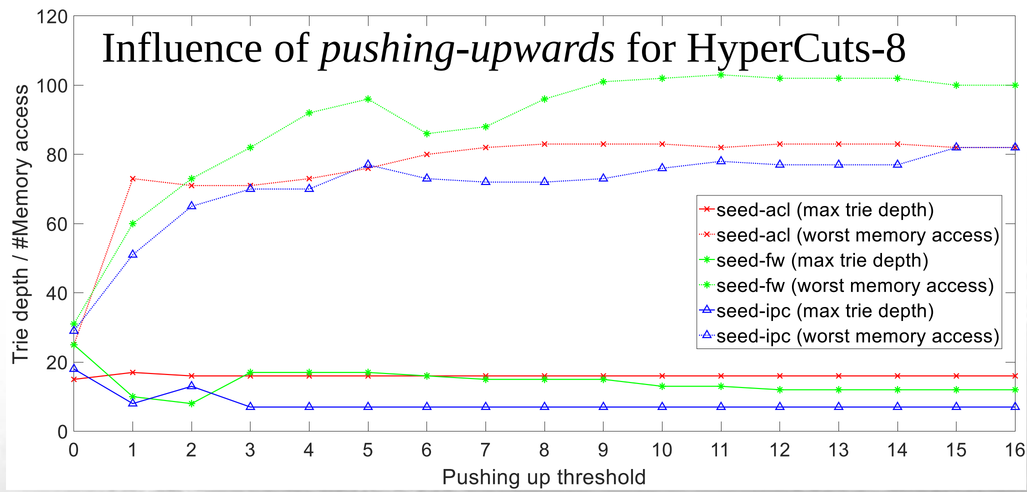
☐ Recent efforts: Effectiveness & Influence

- **Optimization methods**
  - ***pushing-upwards***
  - *rule overlap*
  - *region compaction, etc.*





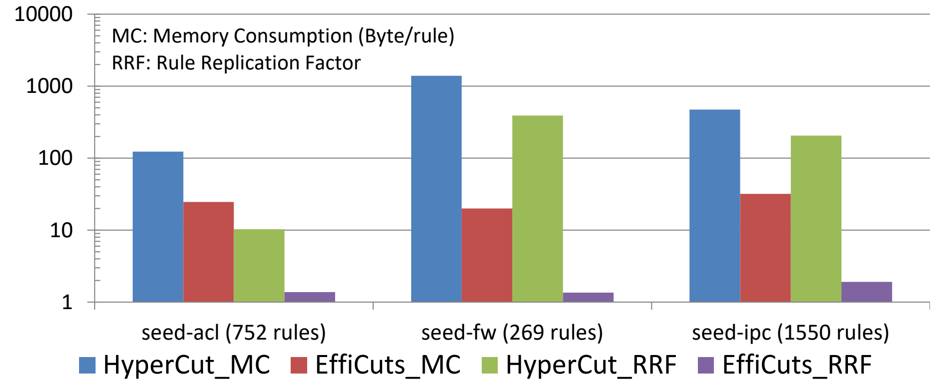




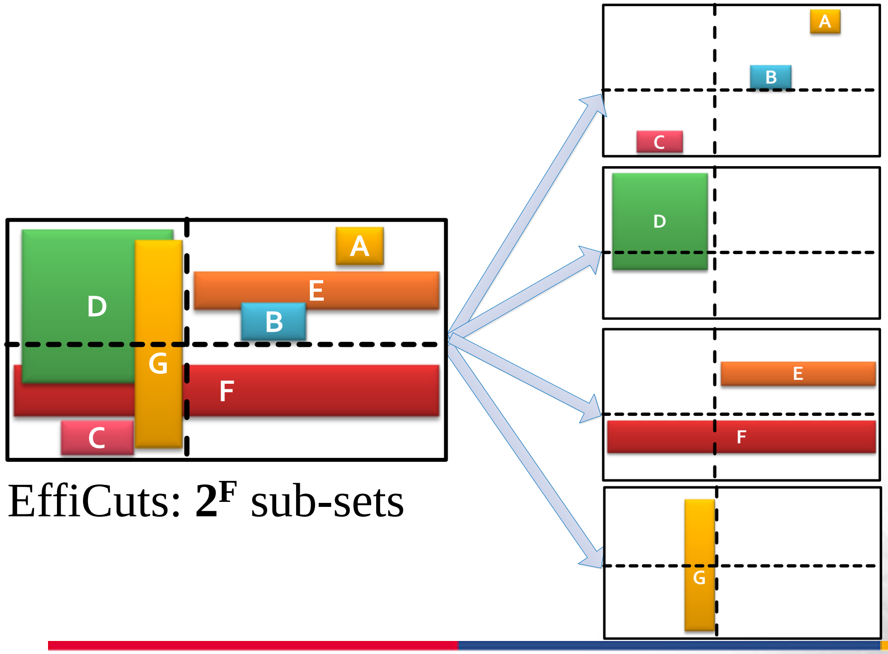
Review & Analysis on Prior Art

Recent efforts: Effectiveness & Influence

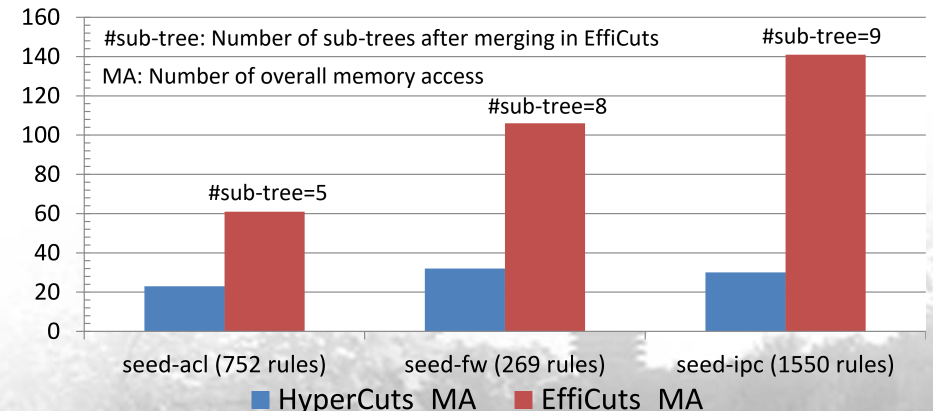
- Optimization methods
- **Rule set partitioning**
  - **all field based: EffiCuts**
  - single field based: HybridCuts
  - IP address based: SmartSplit



Effectiveness of *rule set partitioning* for EffiCuts-8



EffiCuts: $\mathbf{2^F}$ sub-sets



Influence of *rule set partitioning* for EffiCuts-8

Review & Analysis on Prior Art

Recent efforts: Effectiveness & Influence

- Optimization methods
- Rule set partitioning
- **Cutting or Splitting?**
  - EffiCuts: **HyperCuts** + Equi-dense cutting option (i.e., splitting)
  - HybridCuts: One- + multi-dimensional cuttings (i.e., **HyperCuts**)
  - SmartSplit: {**HyperCuts**, HyperSplit} based on *memory estimator*
  - **However**, the performance of these algorithms drop quickly with the size of rule sets increases: Poor scalability of HyperCuts & HyperSplit

Thus, these efforts reduce *rule replications* while sacrificing search or update performance!

Better Solutions?

No optimization method
(with better search & update performance)

+

More scalable rule set partitioning
(with less rule groups)

+

Better combination of Cutting & Splitting
(by exploiting characteristics)

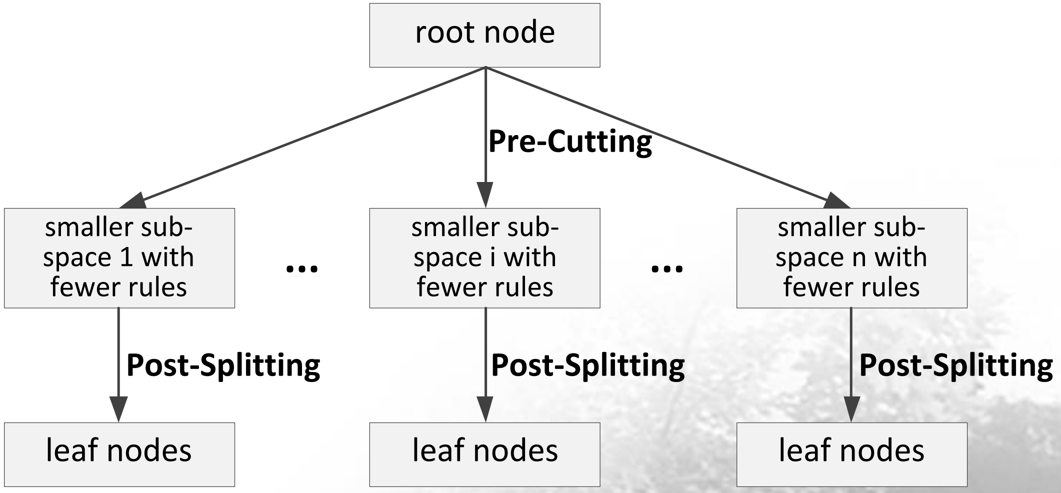
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion

Ideas & Framework

- **Cutting** can separate searching space into **smaller** sub-spaces quickly for faster classification
- **Splitting** can significantly reduce *rule replications* and offer a bounded worst-case search performance for **small** rule sets

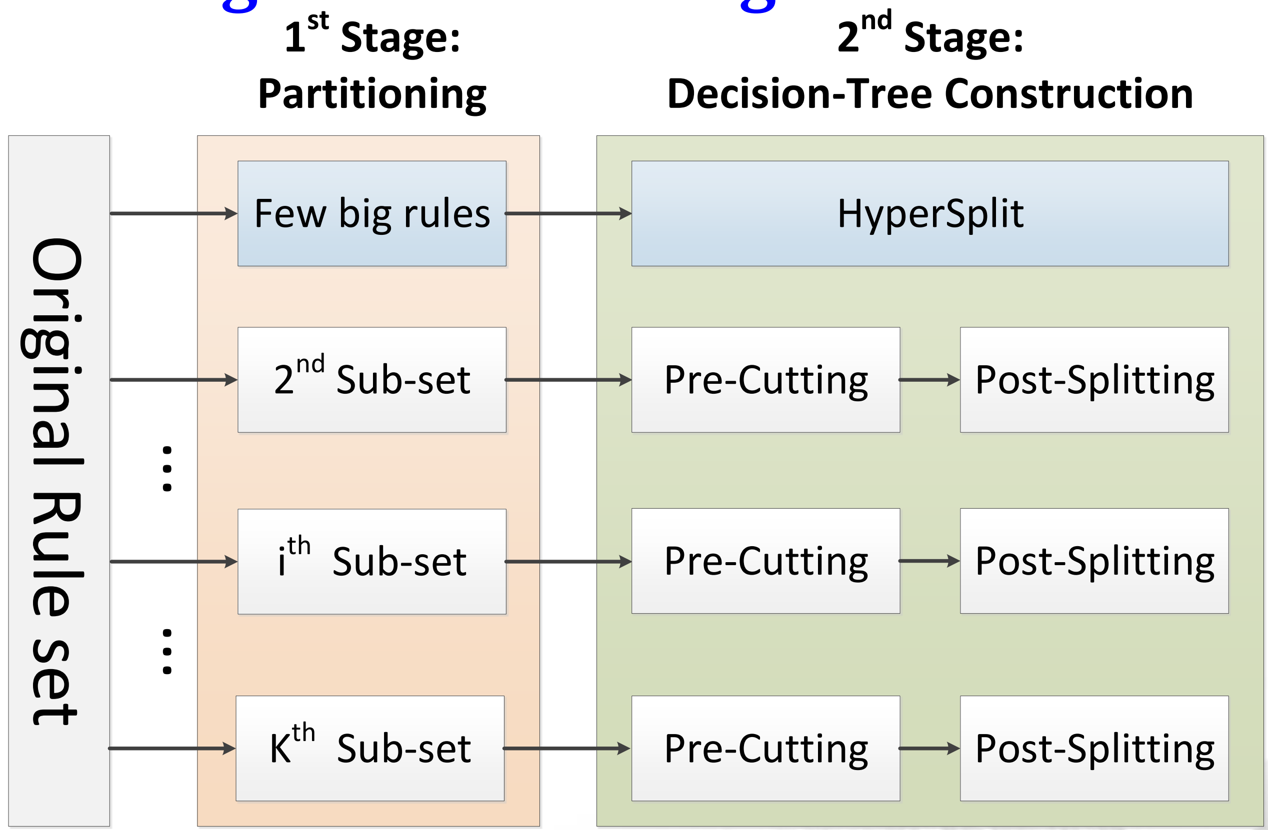
To foster the strengths and circumvent the weaknesses of cutting and splitting, the idea directly perceived is to combine the following two strategies: Faster Pre-Cutting & Explicit Post-Splitting



The framework of CutSplit

More details & Challenges

☐ Preprocessing & Constructing search structure

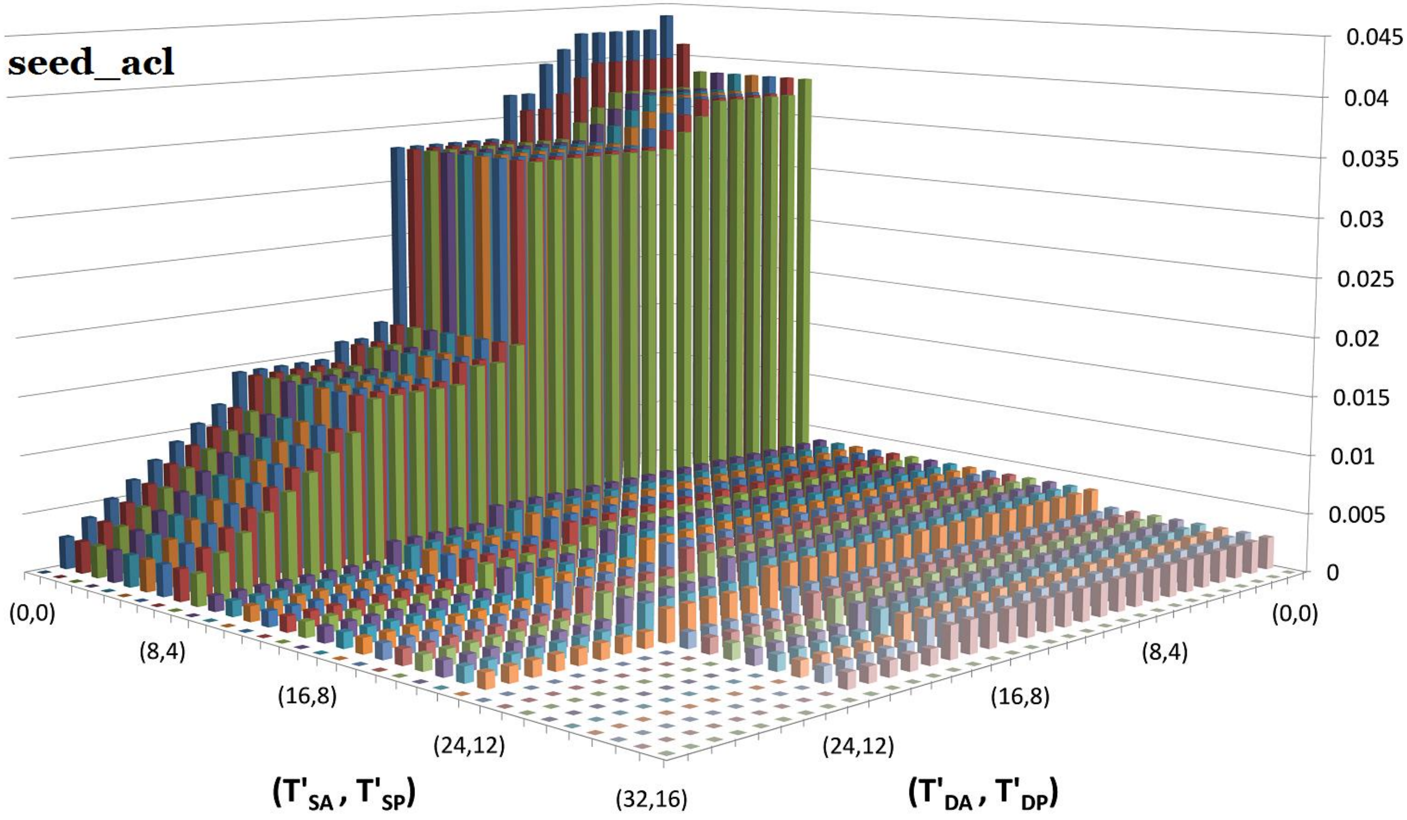


Fewer sub-sets
(Not only for 5-tuple)

No/Fewer rule replication
No optimization in cuttings

Observations (1)

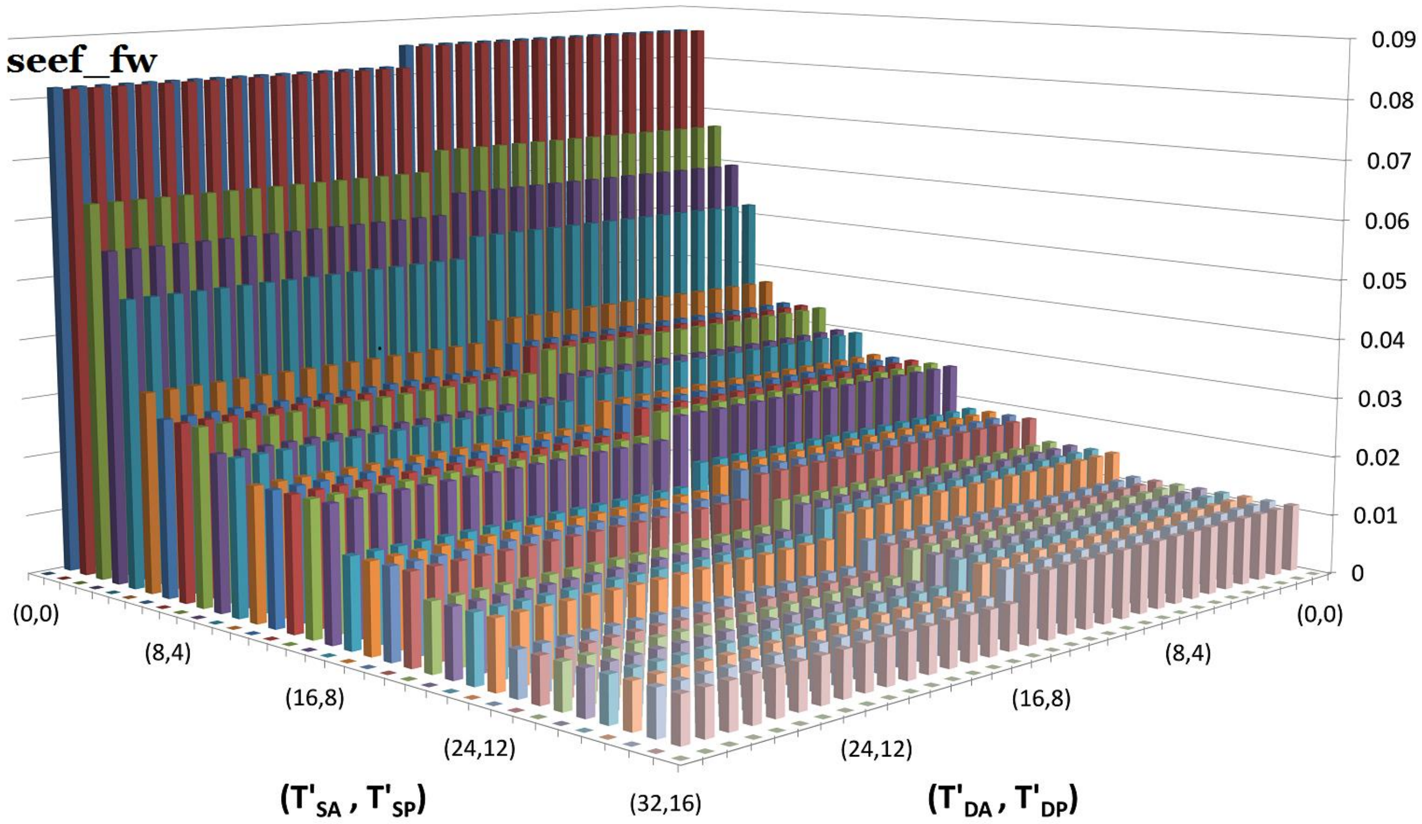
- ☐ **At Least One *Small Field***



The ratio of *big rules* for *seed-acl* rule set

Observations (1)

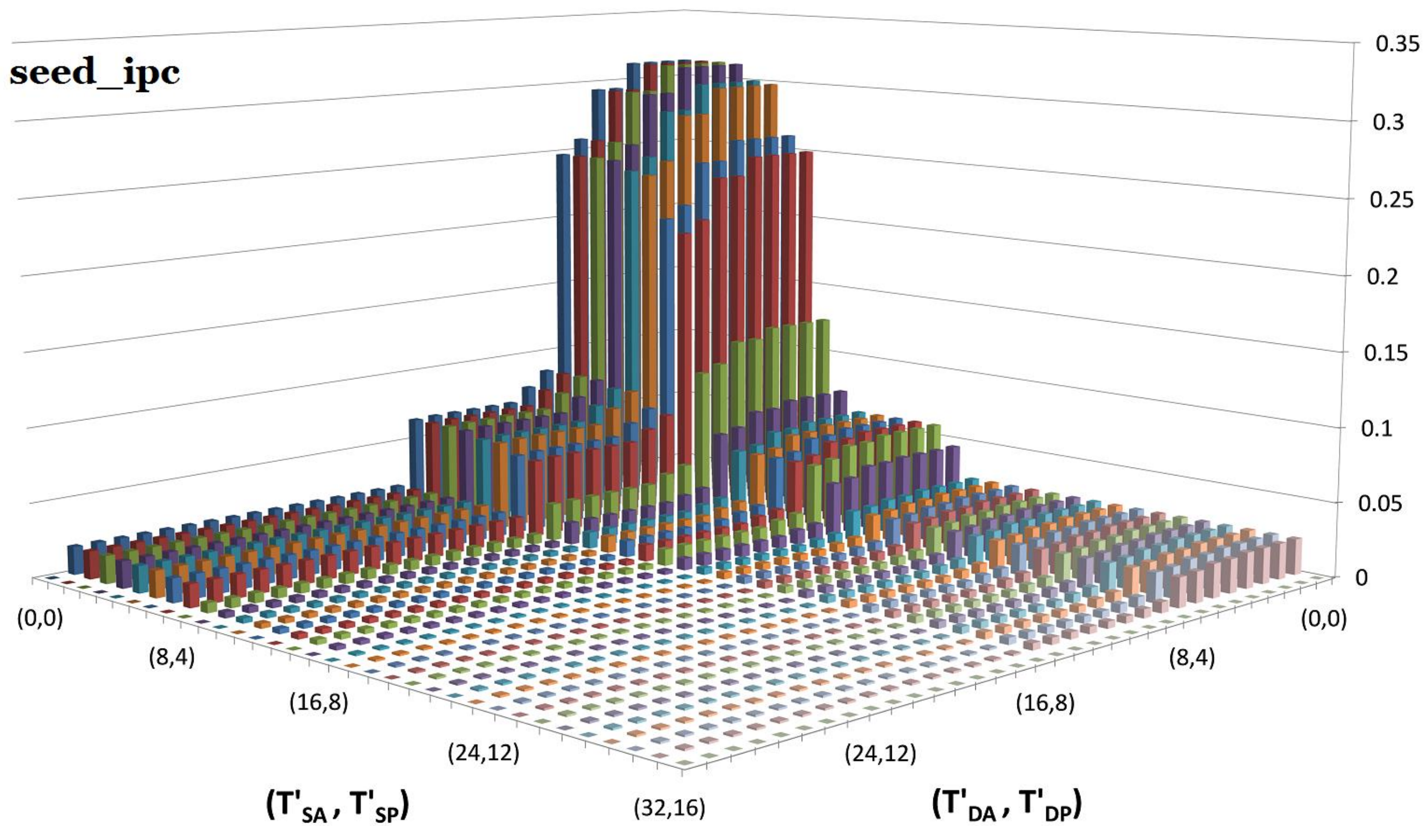
- At Least One *Small Field*



The ratio of *big rules* for *seed-fw* rule set

Observations (1)

- At Least One *Small Field*



The ratio of *big rules* for *seed-ipc* rule set

Observations (2)

☐ Very Few *Small Fields*

Table. Statistical results for 5-tuple & OpenFlow-like rules (Assuming the value of T_i is half of range length in field F_i)

Rule set(#rules)	Number of *big rules*	Number of *small-k* rules				
		k=1	*k=2*	*k=3*	*k=4*	*k⩾5*
seed-acl(752)	3	749	739	425	0	0
seed-fw(269)	4	265	218	17	2	0
seed-ipc(1550)	2	1548	1472	789	5	0
* openflow-1(716)	0	716	708	655	426	0
* openflow-2(864)	0	864	852	761	429	0

Mahalo

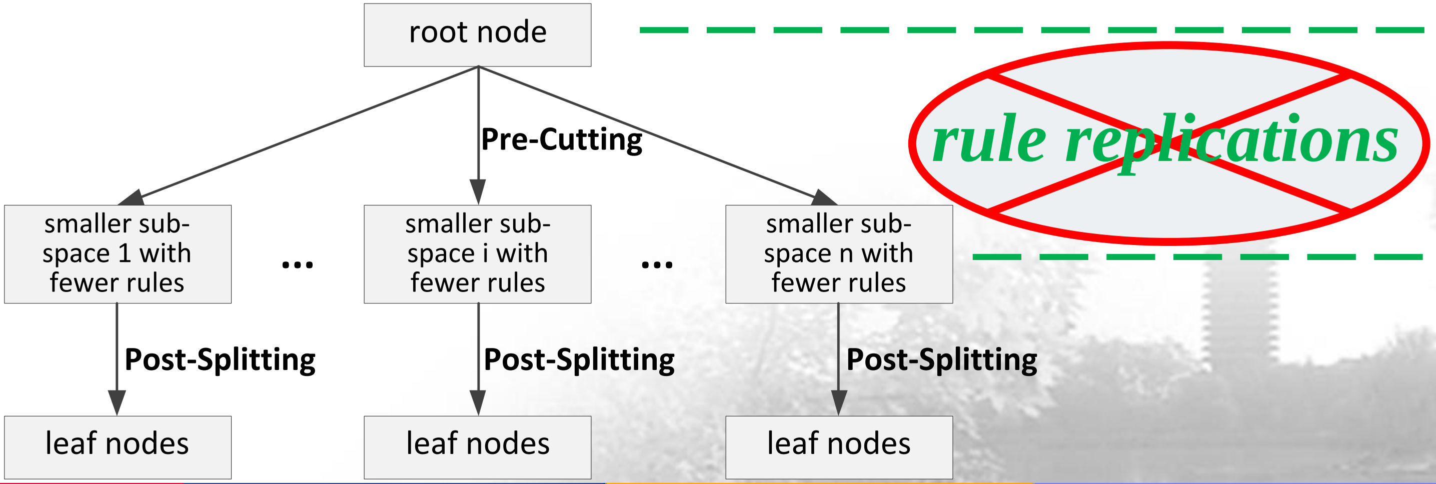
***The two OpenFlow-like rule tables are generated by Tsinghua University, which are based on 216 real-life rules from enterprise customers. We are very grateful to Pro. Jun Li for his selflessness help in this evaluation.**

Scalable Partitioning

- **Step 1:** Remove very few *big rules*
  - HyperSplit for these rules
- **Step 2:** Select a few distinct fields
  - Top-k significant *small fields* (e.g., >95% rules included)
  - Remove remaining rules to *big rules* in Step 1
- **Step 3:** Fields-wise partitioning
  - *M* fields selected for *F*-tuple rule sets → 2^M-1 sub-sets
- **Step 4:** Selective subset merging
  - Sub-set with very few rules → Sub-set with fewer *small fields*

Decision-Tree Constructions

- **CutSplit: Pre-Cutting + Post-Splitting**
  - **Pre-Cutting** on *small fields*
    - Simpler & More efficiently → **No** optimization (e.g., FiCuts)
  - **Post-Splitting** on small sub-sets after cuttings
- **When to switch to Post-Splitting?**
  - Achieving threshold value → **No** rule replication in cutting stage



Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion

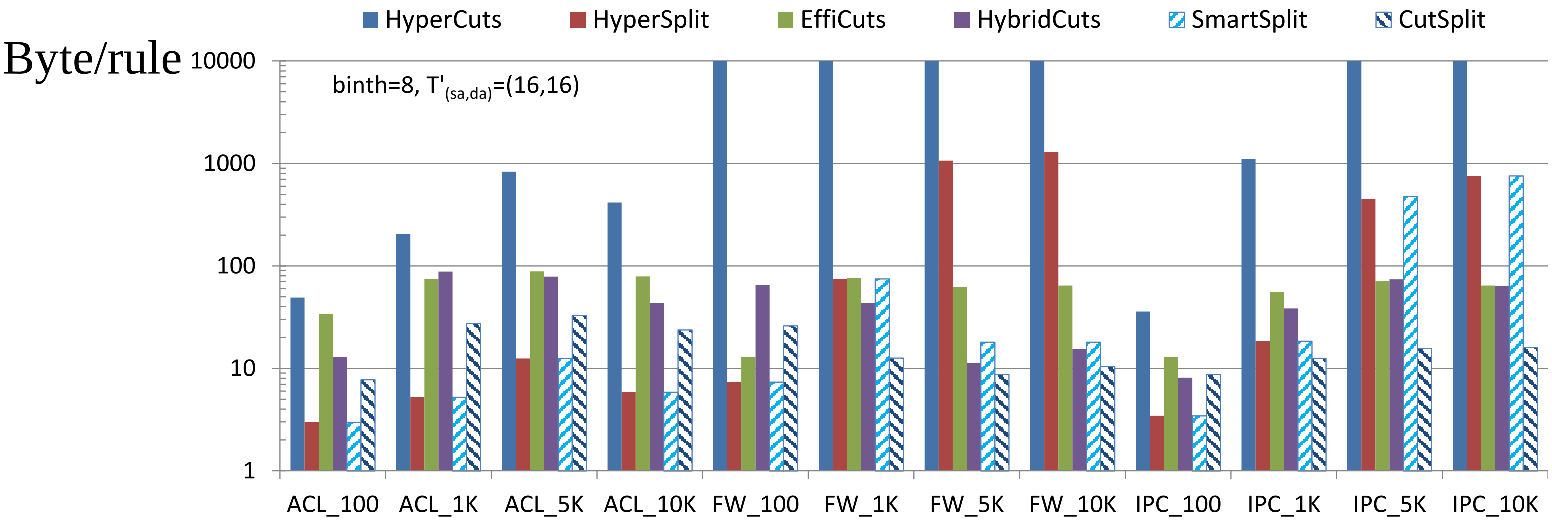
Experimental Setup

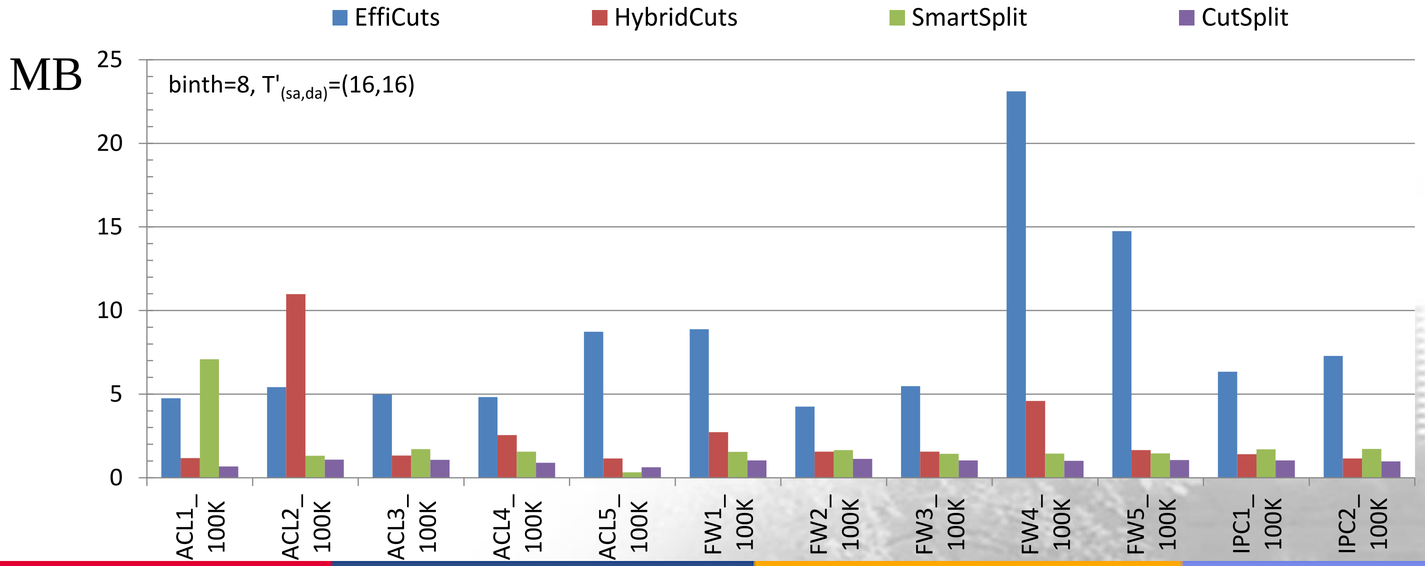
- **Tested with**
  - Publicly available rule sets from Washington University
    - Used the ACL & FW & IPC 100, 1K, 5K, 10K
  - ClassBench
    - Generate ACL & FW & IPC 100K
- **Compared with**
  - Cutting based: HyperCuts, EffiCuts and HybridCuts
  - Splitting based: HyperSplit and SmartSplit
- **Primary metrics**
  - Memory consumption (Decision-tree data structure)
  - Memory accesses
  - Pre-processing time: decision-tree & sub-trees

Mahalo

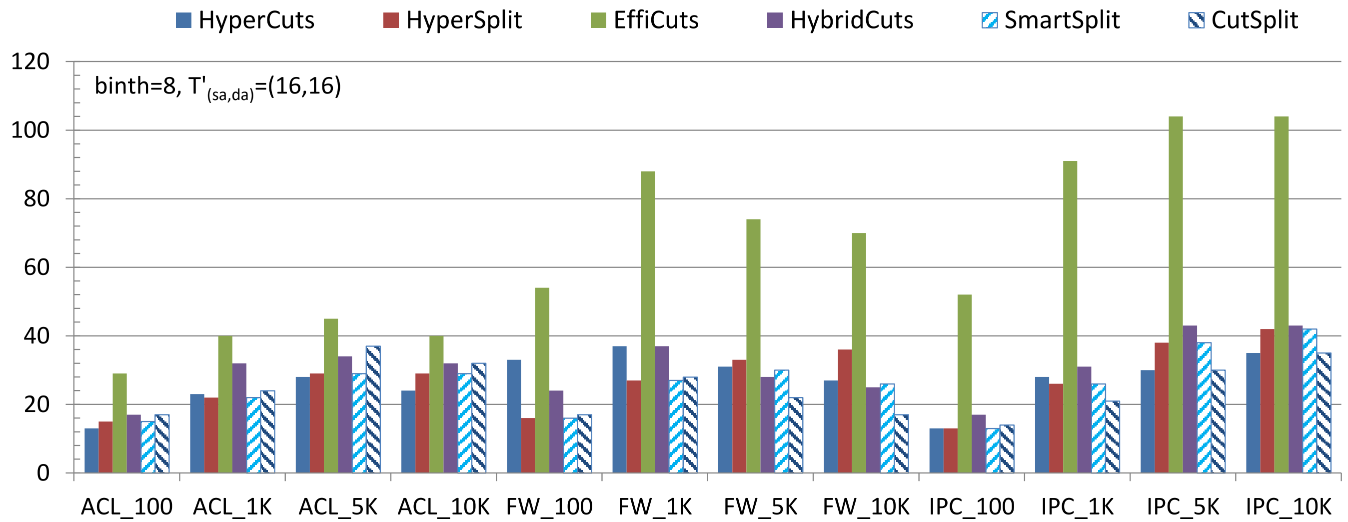
Many thanks to authors of HyperCuts & HyperSplit & EffiCuts for their selflessness help (source codes) in evaluations. As a response, our implementation of CutSplit is publicly available in http://wenjunli.com/CutSplit/

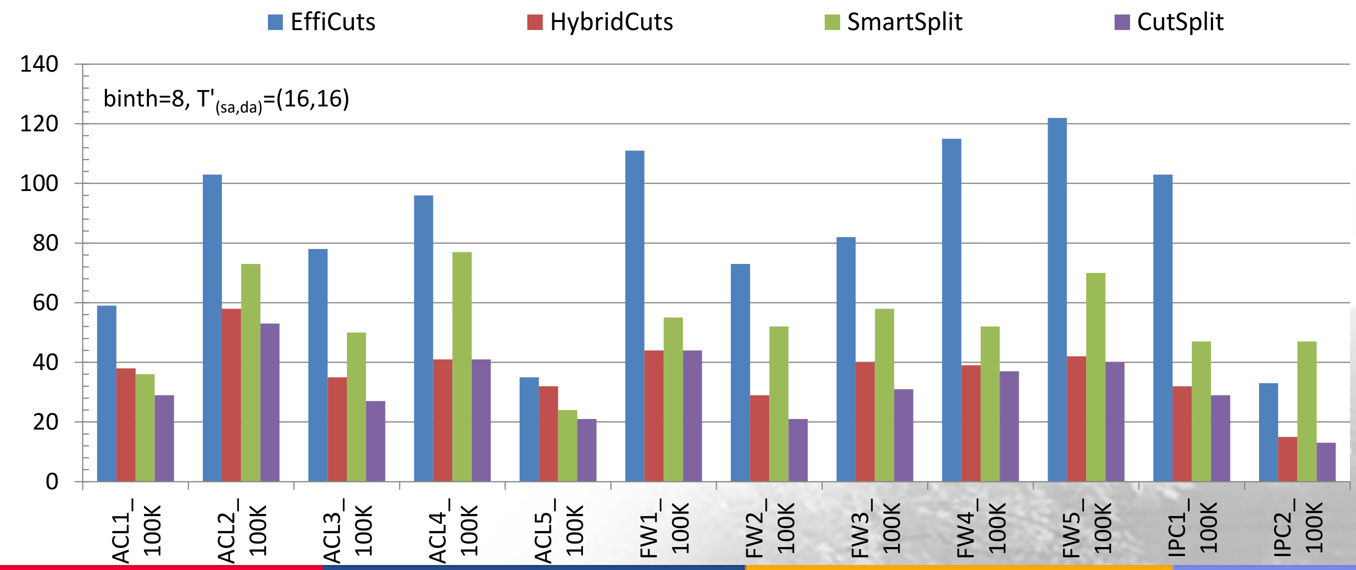
Memory Consumption





Memory Accesses





Pre-processing time: Decision-Tree

Table IV. Pre-processing time for decision-tree construction (s)

Rule set	EffiCuts	HybridCuts	SmartSplit	CutSplit
ACL1_100K	4784.4	183.1	632.5	11.7
ACL2_100K	8338.4	91.0	427.4	4.1
ACL3_100K	8453.6	148.6	6403.7	2.6
ACL4_100K	8232.6	161.8	3336.1	3.4
ACL5_100K	8905.3	138.5	2695.9	3.0
FW1_100K	4250.7	165.1	1392.1	3.0
FW2_100K	2842.2	161.9	1652.9	2.5
FW3_100K	4281.2	187.8	3855.4	3.0
FW4_100K	1662.1	280.3	4553.6	3.5
FW5_100K	3778.4	179.2	3212.7	2.7
IPC1_100K	8615.0	151.5	3133.4	2.6
IPC2_100K	6070.4	229.6	3187.9	2.6
MEAN	5851	173	2874	3.7

Pre-processing time: Sub-trees

Table V. More details about splitting based sub-trees in CutSplit

Rule set	Number of rules		Pre-processing time (us)	
	Worst-case	*Average*	*Worst-case*	*Average*
ACL1_100K	344	17.1	569	45.6
ACL2_100K	473	25.3	6975	125.1
ACL3_100K	31	10.4	207	21.3
ACL4_100K	320	18.7	8693	168.7
ACL5_100K	93	12.8	683	28.9
FW1_100K	193	16.4	2664	71.8
FW2_100K	10	9.4	28	14.8
FW3_100K	118	14.2	1068	43.2
FW4_100K	10	9.0	23	12.9
FW5_100K	111	14.4	869	38.5
IPC1_100K	14	9.7	57	18.1
IPC2_100K	10	9.6	129	15.1
MEAN	144	14	1830	50

Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion

Conclusion

CutSplit:

- In-depth challenge review
- Novel observations
- Scalable partitioning
- Pre-Cutting & Post-Splitting

Future Works

- Determinacy on performance
- Software-hardware combined, e.g., FPGA
- Combine with TSS, TCAM, etc.

Thank you！

Web: http://www.wenjunli.com
E-mail: wenjunli@pku.edu.cn