Network Security - Peter Garamvoelgyi 2018280070

Paper Reading Assignment #1

Multi-Layer Packet Classification with Graphics Processing Units

[1]

1. Introduction

This report is a summary and analysis of the paper *Multi-Layer Packet Classification with Graphics Processing Units* [1].

The report is organized as follows. Section 2 presents a brief overview of the problem of *packet classification*. Section 3 is a discussion of the main insights and contributions of [1]. In Section 4, I present my personal insights into packet classification and academic research in general gained while reading this paper. Finally, Section 5 concludes by giving an overview of the topics discussed.

[1] Image source: www.cigionline.org

2. Packet classification

All information in this chapter is based on [2], the lecture slides, and my personal insights

Packet classification is the process of classifying packets (usually TCP/IP) based on header fields and execute certain actions accordingly. Given a rule-set, we would like to construct a model, that helps us efficiently find the corresponding rule(s) to an incoming packet, based on its headers fiels.

The input packet is usually characterized using the standard 5-tuple of source IP, destination IP, source port, destination port, and protocol (UDP/TCP).

Rules (also called *filters*) contain

- patterns specified using exact values, wildcards, and ranges,
- priorities (often implicitly specified using the order of rules), and
- the associated action (e.g. pass, drop, log, etc.).

Given numerous rules with wildcards, packet classification can easily become the bottleneck of packet processing. However, by using appropriate data structures and leveraging characteristics of real-world filter sets, it is possible to achieve the desirable average performance. It is also common to use hybrid hardware-software solutions to achieve high performance.

Packet classifications algorithms can be grouped into four different categories:

1. **Exhaustive search**: Searching through all the entries.

   The two extreme approacheas are linear search and TCAM[2]. The first is easy to implement but it has suboptimal time and storage complexity. The latter is much faster but suffers from high price and scaling issues.

2. **Decision tree**: Structure rules in trees/tries of various complexity.

   Well-known approaches include Grid-of-Tries [3], HiCuts [4], HyperCuts [5], HyperSplit, E-TCAM, and FIS trees [6]. The performance of these algorithms varies

[2] TCAM: Ternary Content Addressable Memory

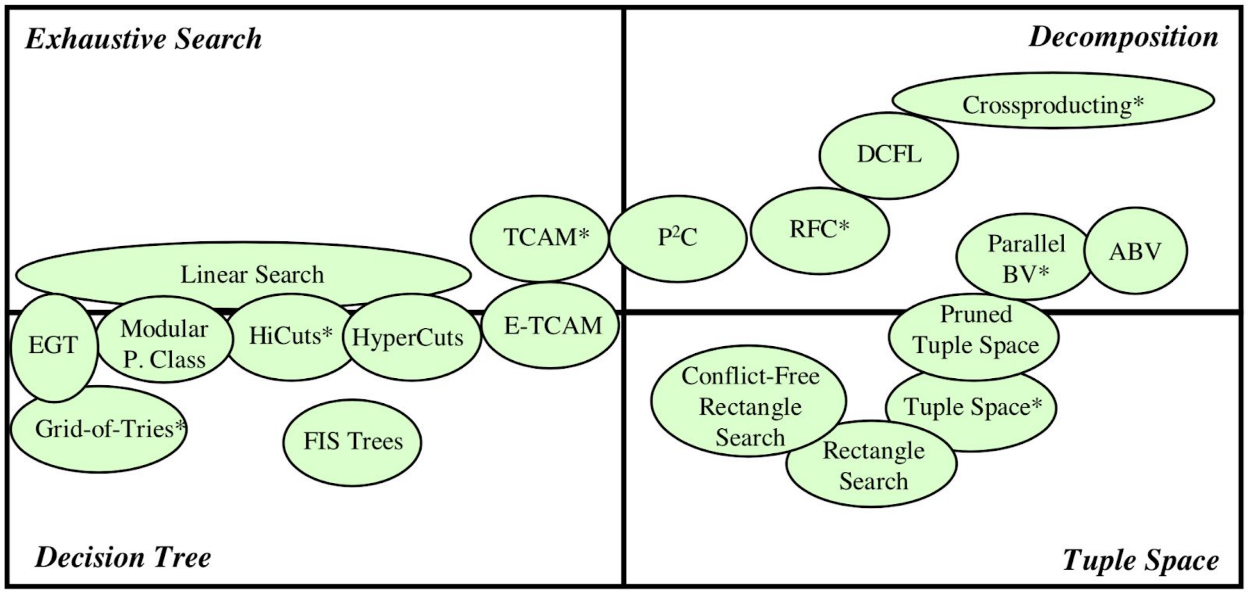
significantly. Many of them treat the rules as a d-dimensional space and attempt to segment this space by finding good cuts based on various heuristics.

3. **Decomposition**: Perform multiple single-field searches.

   Decomposition-based approaches execute high performance single-field algorithms in a parallel fashion and then proceed to combine the results. Examples include Parallel Bit-Vectors (BV) [7], ABV [8], AFBV, Crossproducting [3], and RFC [9].

4. **Tuple space**: Partition filter set by tuples.

   Tuple space approaches tend to make heavy use of hash tables. While having suboptimal complexity, these methods usually support incremental updates, making them popular in software switches like Open vSwitch.



The key takeaways we can gain by studying these approaches are

1. Designing algorithms like these include making explicit tradeoffs.
2. Using real datasets can help improve average performance.
3. Utilizing graphical view of the problem can help come up with unintuitive solutions.

Recurrent themes include runtime and storage performance, parallelizability, caching, hardware costs, and the possibility of dynamic updates.

3. Paper overview

All information in this chapter is based on [1] and my personal insights

The authors of the paper *Multi-Layer Packet Classification with Graphics Processing Units* tackle a well-defined problem. The mainstream adoption of virtualization and SDN[3] brought about the renaissance of software-based virtual switches like Open vSwitch and Cisco Nexus 1000V. An example application is the use of such switches in hypervisors to enable communication between virtual machines in data centers.

Software-based packet classification has some unique challenges like large, dynamic rule tables and multidimensional tuples. These challenges make existing approaches suboptimal. Using general-purpose CPUs, packet classification in software switches has become a bottleneck. According to the authors, highly parallel GPU-based packet classification solves this issue and moves the bottleneck back to high-speed packet IO.

The paper has two main contributions:

1. Efficient, GPU-accelerated implementation of three packet classification algorithms: linear search, tuple search, and bloom search. The latter one is also introduced in this paper for the first time.
2. Implementation and evaluation of GSwitch, a software switch using the above GPU-based algorithms.

The authors utilize their domain knowledge of GPGPU programming[4] to achieve state-of-the-art performance. The applied methods include

- *Coalesced memory access* for reducing latency. Using the right access patterns, multiple memory accesses can be served with a single physical read.
- Heavy local *caching* utilizing on-chip SRAM and memory sharing.
- 100% utilization of GPU cores by choosing the right parameters and work allocation strategies.

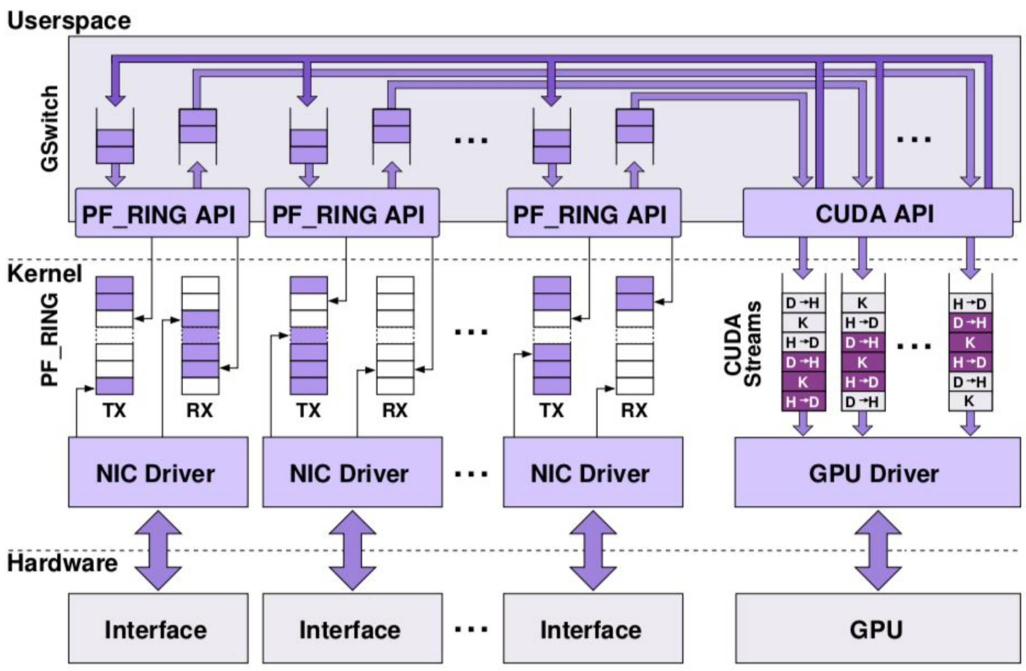
[3] SDN: Software Defined Networking
[4] GPGPU: General Purpose GPU programming

Another novel insight is the use of *bloom filters* for preventing unnecessary table lookups. A bloom filter is a data structure that allows O(1) detection of negative matches, i.e. when a class of rules (defined by a mask) does not match a tuple. It does give false positives, but the rate of these is adjustable with certain filter parameters.

The paper describes multiple design decisions including

- What to copy to the GPU and what not to?
- How to split work among several processing units?
- How to access memory?
- What is the right hash function to use?

Furthermore, the architecture of GSwitch is also presented, giving valuable insights into the challenges of high-performance packet engines. PF_RING is used for high-performance packet I/O with separate threads and buffers for each network interface. To increase performance, certain CUDA-specific features are also employed (e.g. streams). This design shows the expertise of the authors in network programming, low-level systems programming, and GPU programming.



The authors used a test setup of a machine with a single CPU and a single GPU of comparable price to evaluate the proposed approaches using multiple configurations and benchmarks. According to their results, applying GPUs bring *"up to 7x (linear search), 11x (tuple search), 12x (bloom search)"* speedup for packet classification. GSwitch can serve *"64-byte frames at 30 Gbps and a maximum per-packet latency of 500 μs"*.

4. Insights and ideas

In this section I am going to discuss some of the insights I gained by reading the paper.

First, being a well-written document with significant contributions to a well-defined area of study, this paper gave me useful insight into the study of packet classification and the nature of academic research in general.

- **Algorithm engineering**: Once I had a class called *algorithm engineering* where the teacher argued that most of the work associated with algorithms is *"improving the constant multipliers and applying domain-specific knowledge to improve performance."*

  It seems to me that most of academic research into algorithmic problems is similar: breakthroughs and asymptotically better solutions occur, but most of the work is about **a)** non-asymptotic performance improvements and **b)** considering tradeoffs and exploiting domain-specific knowledge.

  An example for **a)** in this paper is the use of bloom filters for eliminating unnecessary table lookups. While the worst case complexity remains the same, this methods significantly improves time complexity in real-world scenarios.

  Another example is caching and other hardware-aware optimization, like coalesced memory access and parallelization. These methods tend to give great performance improvements but they are very hard to represent in mathematical analysis.

  An example for **b)** is the detailed parameter evaluation presented in the paper and the use of both synthetic and real-world rule sets for benchmarking.

- **Comparison with related work**: I really liked how the authors compared their approach with existing solutions explicitly in the *related works* section. In papers I read before this was either omitted or not elaborated so clearly.

  Another takeaway from this section is that comparison of algorithms and solutions is hard. Even for papers from the same research area, the authors might use drastically different evaluation methods and setups. Moreover, the implementations of many algorithms are not publicly available due to various reasons. Given these challenges, it is noteworthy that the authors still managed to give a rough comparison with a related paper.

  A solution to the problem of reliable comparisons is to create standardized[5] testing frameworks and benchmarks. For packet classification, such benchmarks (rule sets) exists, but creating a testing framework is still an open issue as far as I know.

- **What to explain**: A non-obvious question is: What background knowledge is expected from the reader and what should be explained in a paper?

  In this paper, the authors knew that they cannot expect experts of packet classification algorithms to know all the intricacies of GPU programming, so they devoted a whole section to explaining this.

- **Experimental setup and evaluation is key**: Progress in science and engineering relies on accurate, reproducible, and comparable results from researchers.

  For getting accurate results, it is crucial to choose the right performance metrics. Ignoring some key metrics might result in solutions that violate real-world constraints. For instance, given unlimited storage, it is theoretically possible to create O(1) packet classification algorithms by using hash tables, but this approach lacks practical applicability.

  For reproducibility and comparability, it is important to give a detailed description of the assumptions, the hardware setup, and the test scenarios.

[5] Caveat: https://xkcd.com/927

Second, below are some aspects where I felt there is room for improvement.

- **Misleading wording**: The authors often used terms like *"up to 7x speedup"*.

  While this wording is technically not incorrect, it can be misleading. For example, if there is a 7x speedup for 1% of the inputs and only 2x speedup for the rest, this wording could be used to give the reader the impressions of a larger improvement.

- **Related works**: I felt that the *related works* section is suspiciously short, it only mentions two papers. This is probably due to the novelty of the approach, in which case it is acceptable. In most papers, however, it is advisable to have a more thorough evaluation of related approaches.

Third, I will share some ideas for future research in this topic.

- **Scaling**: The authors' evaluations are based on one specific hardware setup using a single GPU. It would be interesting to see how the performance scales with a) the performance of the GPU and b) the number of GPUs.

  When increasing the number of GPUs, it is likely that some new challenges emerge, e.g. job allocation, merging results, etc. However, most of these have been investigated in previously, so the main question is how to apply the solutions.

- **Deployment in real-world settings**: While not necessarily academic in nature, it would be interesting to see the proposed approach deployed in data centers and SDN solutions.
- **Bloom filters**: Using efficient bloom filters to eliminate unnecessary lookups is a great idea. It would be interesting to see if the same approach is applicable to other package classification algorithms.

5. Conclusion

The assigned paper [1] has multiple important contributions to the research area of packet classification. By demonstrating the practical feasibility of utilizing GPUs for this task and showing the performance benefits, the authors opened up several new research directions. The paper was also a good example for me on what constitutes a well-written publication.

References

[1] Varvello, Matteo, Rafael Laufer, Feixiong Zhang, and T. V. Lakshman. "Multilayer packet classification with graphics processing units." IEEE/ACM Transactions on Networking 24, no. 5 (2016): 2728-2741.

[2] D. E. Taylor, "Survey & Taxonomy of Packet Classification Techniques," Technical Report WUCSE-2004-24, Washington University in Saint-Louis (WUSTL), 2004.

[3] V. Srinivasan, S. Suri, G. Varghese, and M. Waldvogel, “Fast and Scalable Layer Four Switching,” in ACM Sigcomm, June 1998.

[4] P. Gupta and N. McKeown, “Packet Classification using Hierarchical Intelligent Cuttings,” in Hot Interconnects VII, August 1999.

[5] S. Singh, F. Baboescu, G. Varghese, and J. Wang, “Packet Classification Using Multidimensional Cutting,” in Proceedings of ACM SIGCOMM’03, August 2003. Karlsruhe, Germany.

[6] A. Feldmann and S. Muthukrishnan, “Tradeoffs for Packet Classification,” in IEEE Infocom, March 2000.

[7] T. V. Lakshman and D. Stiliadis, “High-Speed Policy-based Packet Forwarding Using Efficient Multi- dimensional Range Matching,” in ACM SIGCOMM‘98, September 1998.

[8] F. Baboescu and G. Varghese, “Scalable Packet Classification,” in ACM Sigcomm, August 2001.

[9] P. Gupta and N. McKeown, “Packet Classification on Multiple Fields,” in ACM Sigcomm, August 1999.