PacketMill: Toward per-core 100-Gbps Networking
Extended Abstract

Alireza Farshin[1], Tom Barbette[1], Amir Roozbeh[1,2], Gerald Q. Maguire Jr.[1], and Dejan Kostić[1]

[1]KTH Royal Institute of Technology, [2]Ericsson Research

We present PacketMill, a system for optimizing software packet processing, which (i) introduces a new model to efficiently manage packet metadata *and* (ii) employs code-optimization techniques to better utilize commodity hardware. PacketMill grinds the whole packet processing stack, from the high-level network function configuration file to the low-level userspace network (specifically DPDK) drivers, to mitigate inefficiencies and produce a customized binary for a given network function. Our evaluation results show that PacketMill increases throughput (up to 36.4 Gbps – 70%) & reduces latency (up to 101 µs – 28%) and enables nontrivial packet processing (e.g., router) at ≈100 Gbps, when new packets arrive $>10\times$ faster than main memory access times, while using only *one* processing core.

1. Motivation

Networking has shifted from inflexible, proprietary, and specialized hardware toward Software-defined Networking (SDN) and Network Functions Virtualization (NFV). Today many network appliances are realized using commodity hardware and the network functions are increasingly software-driven. The flexibility and programmability of such platforms has led to many software networking solutions (such as Open vSwitch (OVS) [17], Click-based frameworks [16, 1, 9], BESS [11, 10], and Vector Packet Processing (VPP) [8]). Unfortunately, the introduction of multi-hundred-gigabit network equipment and dramatic increases in telecommunication network bandwidth strain the performance of commodity hardware [14], due to the demise of Moore's law and Dennard scaling putting a cap on commodity hardware's performance [5]. Realizing 100-Gbps networking is extremely challenging, as the time budget for processing small packets, i.e., 6.72 ns to process a 64-B packet before receiving the next one makes *nanosecond level savings* count. Consequently, software needs to operate in the L1 & L2 caches – minimizing even Last Level Cache (LLC) accesses [19, 6, 7, 18].

While many have tried to introduce in-network processing via hardware (e.g., P4 architecture [2] and modern/programmable Network Interface Card (NIC)) to address the performance limitations [21]; today, many network functions are deployed on commodity hardware, via *unspecialized* modular software with software-based packet processing by, for example, Ericsson, Cisco, and Intel [4, 13, 20]. Unfortunately, the software-driven networking solutions come at the price of lower performance due to (i) code inefficiency, mainly coming from generality and modularity of networking frameworks, and (ii) poor metadata management.

Our objective is to produce an optimized binary while maintaining high-level modularity and flexibility, as opposed to relying on handwritten assembly code [15]. This paper shows that efficient metadata management (i.e., specialization of Data Plane Development Kit (DPDK)'s buffers) and employing code optimizations (to minimize unnecessary memory accesses, improve cache locality, etc.) facilitates realizing our goal of software-based packet processing at 100 Gbps on commodity hardware.

2. Limitations of the State of the Art

We employ two techniques to optimize the performance of software-based packet processing: (i) efficient metadata management and (ii) code-optimization techniques. To the best of our knowledge, the former has *not* explicitly been examined/optimized before. Examining this unexplored territory enables a *single* core to forward packets >100 Gbps. The latter has been partially tackled by a similar approach to improve performance via code optimizations: ① The most relevant is the work of Kohler et al. [12] who proposed an optimization toolkit for the Click modular router [16] to reduce inefficiencies of a Network Function (NF) based on its input configuration. This work motivated us to further mitigate code inefficiencies in software packet processing frameworks to exploit the availability of: (i) multi-hundred-Gbps network interfaces and (ii) programming language (PL) tools (e.g., LLVM toolchain). To do so, we resurrected their tool, called `click-devirtualize`, and extended it with new optimizations. Additionally, we use LLVM's optimization passes to apply intermediate representation (IR)-code modifications. ② Bangwen et al. [3] used classic compiler optimization techniques in combination with symbolic execution to filter out infeasible paths of specialized networking software (e.g., snort). Their main focus was protocol mismatches (occurring between the development and deployment phases). While some of our techniques are similar, we focus on mitigating inefficiencies induced by modularity in general-purpose packet processing frameworks versus eliminating redundant logic.

3. Key Insights

Our main insight is that efficient packet processing at 100-Gbps calls for holistic system optimization, specifically *milling* the entire software stack to *squeeze* every bit of performance from the hardware. We are the first to (i) empirically examine/optimize metadata management models for packet processing and (ii) advocate low-level optimizations to process packets at near-100-Gbps rates with a *single* core. Our techniques are geared toward making sure that the software can run in the fastest (L1/L2) caches.

4. Main Artifacts

We design, build, and evaluate a system, called PacketMill, to optimize the performance of a popular modular framework used for composing complex network functions on top of commodity hardware. We propose a new metadata management model, called X-Change, that realizes customized buffers when using DPDK, rather than relying on the generic `rte_mbuf` structure. Additionally, we propose a set of common & uncommon code optimizations to (i) the source code and (ii) the IR code while employing link-time optimization (LTO) techniques. PacketMill exploits the information defining a NF to mitigate virtual calls, improve constant propagation & constant folding, and reorder commonly used data structures in modular packet processing.
Implementation. PacketMill is composed of three main components:

1. **X-Change**: developed as an Application Programming Interface (API) within DPDK,
2. **source-code modifications**: implemented on top of a resurrected & modified `click-devirtualize`, and
3. **IR-based modifications**: implemented as LLVM optimization passes applied to the *complete* program's IR bitcode as extracted from LTO.

Evaluation. We apply PacketMill to DPDK-based FastClick [1], where we composed five different NFs: a simple forwarder, a router, a Intrusion Detection System (IDS) followed by a router, a Network Address Translation (NAT), and a synthetic memory- & compute-intensive NF. We evaluate our proposed metadata management model (X-Change) and compare it with two common *de facto* techniques (i.e., copying & overlaying) used in other packet processing frameworks (i.e., BESS & VPP) re-implemented in FastClick. Additionally, we use a real campus trace to demonstrate the effectiveness of PacketMill to improve per-core packet processing at 100 Gbps. Although we focus on optimizing a specific framework (i.e., FastClick), our results and techniques should be useful in other performance-sensitive contexts (all of those concerned with nanosecond- and microsecond-level improvements). The paper (§4) discusses the benefits of PacketMill when NFs are running at different scenarios (e.g., various frequencies & #cores).

5. Key Results and Contributions

Our results demonstrate that PacketMill improves **both** microarchitectural metrics (i.e., reducing cache misses) and application-level metrics (i.e., decreasing latency and increasing network throughput) when running at 100 Gbps.

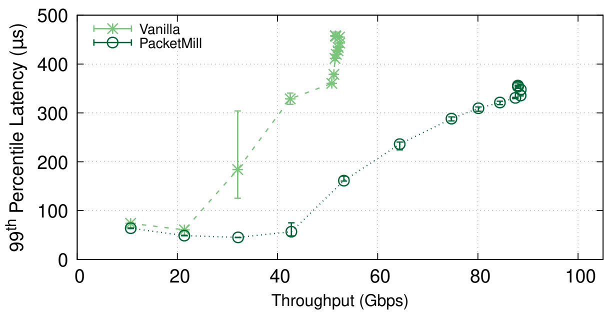

Figure 1: PacketMill improves per-core packet processing. Overlapped markers show that the performance can be capped despite increasing the offered load.

Figure 1 demonstrates that PacketMill improves the packet processing at 100 Gbps when a router running in a *single core* is forwarding packets. More specifically, our proposal shifts the knee of the tail latency vs. throughput curve, to achieve lower latency *even* when the load is higher.
Contributions. In this paper, we:

- Highlight the importance of metadata management in packet processing and propose a new model, called X-Change, to mitigate inefficiencies,
- Design & implement PacketMill to optimize the performance of packet processing frameworks via low-level optimization,
- Demonstrate the importance of employing code optimization & efficient metadata management to operate at >100 Gbps rates.
- Extend DPDK's build system to employ LTO via Clang and produce LLVM IR bitcode,
- Additionally, we plan to release our source code and the scripts used with the Network Performance Framework (NPF) tool to facilitate others reproducing our results.

6. Why ASPLOS

This paper advocates the importance of employing low-level optimization techniques to make it possible to process packets at 100 Gbps. To do so, our paper combines two main research areas, i.e., off-chip networking *and* compiler techniques & optimizations. More specifically, we mechanically produce a specialized binary from general-purpose software packet processing frameworks. We believe ASPLOS is a perfect venue for our paper, as we exploit the synergy of two *system* areas, as encouraged by the ASPLOS CFP.

7. Citation for Most Influential Paper Award

This paper was the first work emphasizing full software stack optimization for multi-hundred-gigabit networking. It illustrated that per-core 100-Gbps networking requires the software to operate in the high-level cache memories (i.e., L1 & L2 caches). This was done by meticulously managing metadata and carefully optimizing the binary of networking applications.

References

[1] Tom Barbette, Cyril Soldani, and Laurent Mathy. Fast Userspace Packet Processing. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '15, pages 5–16, Washington, DC, USA, 2015. IEEE Computer Society.

[2] 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.

[3] Bangwen Deng, Wenfei Wu, and Linhai Song. Redundant Logic Elimination in Network Functions. In *Proceedings of the Symposium on SDN Research*, SOSR '20, page 34–40, New York, NY, USA, 2020. Association for Computing Machinery.

[4] Ericsson. Supercharging the Evolved Packet Gateway. Technical report, Ericsson, June 2017. https://www.ericsson.com/assets/local/digital-services/doc/Supercharging-the-Evolved-Packet-Gateway.pdf, accessed 2020-07-24.

[5] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, June 2011.

[6] Alireza Farshin, Amir Roozbeh, Gerald Q. Maguire Jr., and Dejan Kostić. Make the Most out of Last Level Cache in Intel Processors. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, pages 8:1–8:17, New York, NY, USA, 2019. ACM.

[7] Alireza Farshin, Amir Roozbeh, Gerald Q. Maguire Jr., and Dejan Kostić. Reexamining Direct Cache Access to Optimize I/O Intensive Applications for Multi-hundred-gigabit Networks. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 673–689. USENIX Association, July 2020.

[8] FD.io. Vector Packet Processing - One Terabit Software Router on Intel Xeon Scalable Processor Family Server. Technical report, Cisco, Intel Corporation, FD.io, July 2017. https://fd.io/docs/whitepapers/FDioVPPwhitepaperJuly2017.pdf, accessed 2020-07-24.

[9] Massimo Gallo and Rafael Laufer. ClickNF: a Modular Stack for Custom Network Functions. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 745–757, Boston, MA, July 2018. USENIX Association.

[10] Sangjin Han, Keon Jang, Aurojit Panda, Shoumik Palkar, Dongsu Han, and Sylvia Ratnasamy. Berkeley Extensible Software Switch (BESS), 2015. http://span.cs.berkeley.edu/bess.html, accessed 2020-07-22.

[11] Sangjin Han, Keon Jang, Aurojit Panda, Shoumik Palkar, Dongsu Han, and Sylvia Ratnasamy. SoftNIC: A Software NIC to Augment Hardware. Technical Report UCB/EECS-2015-155, EECS Department, University of California, Berkeley, May 2015.

[12] Eddie Kohler, Robert Morris, and Benjie Chen. Programming Language Optimizations for Modular Router Configurations. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS X, page 251–263, New York, NY, USA, 2002. Association for Computing Machinery.

[13] Maciek Konstantynowicz, Patrick Lu, and Shrikant M. Shah. Benchmarking and Analysis of Software Data Planes. Technical report, Cisco, Intel Corporation, FD.io, Dec 2017. https://fd.io/wp-content/uploads/sites/34/2018/01/performance_analysis_sw_data_planes_dec21_2017.pdf, accessed 2019-07-24.

[14] Niall McDonnell and Gage Eads. Queue Management and Load Balancing on Intel Architecture, 2020. https://tinyurl.com/yxv9cgpj, accessed 2020-08-08.

[15] László Molnár, Gergely Pongrácz, Gábor Enyedi, Zoltán Lajos Kis, Levente Csikor, Ferenc Juhász, Attila Kőrösi, and Gábor Rétvári. Dataplane Specialization for High-Performance OpenFlow Software Switching. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 539–552, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The Click Modular Router. In *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, SOSP '99, page 217–231, New York, NY, USA, 1999. Association for Computing Machinery.

[17] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. The Design and Implementation of Open vSwitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA, May 2015. USENIX Association.

[18] Shelby Thomas, Rob McGuinness, Geoffrey M. Voelker, and George Porter. Dark Packets and the End of Network Scaling. In *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*, ANCS '18, pages 1–14, New York, NY, USA, 2018. ACM.

[19] Shelby Thomas, Geoffrey M. Voelker, and George Porter. CacheCloud: Towards Speed-of-light Datacenter Communication. In *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, Boston, MA, July 2018. USENIX Association.

[20] Georgii Tkachuk, Maciek Konstantynowicz, and Shrikant M. Shah. Benchmarking Software Data Planes - Intel Xeon Skylake vs. Broadwell. Technical report, Cisco, Intel Corporation, FD.io, March 2019. https://www.lfnetworking.org/wp-content/uploads/sites/55/2019/03/benchmarking_sw_data_planes_skx_bdx_mar07_2019.pdf, accessed 2020-07-24.

[21] Yuta Tokusashi, Huynh Tu Dang, Fernando Pedone, Robert Soulé, and Noa Zilberman. The Case For In-Network Computing On Demand. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, pages 21:1–21:16, New York, NY, USA, 2019. ACM.