Effects of Unaligned Memory Access on the Performance of Packet Analyzers

Maximilian Pudelko, Paul Emmerich, Georg Carle
Technical University of Munich Departments of Informatics
Chair of Network Architectures and Services
{pudelko|emmericp|carle}@in.tum.de

ABSTRACT

In this paper we verify that the layout of packets in memory can have beneficial as well as detrimental effects on the performance of packet analyzers. We develop and run a benchmark to quantify these effects on a sample of different CPUs. We report performance differences of up to 27% caused by the alignment of packet data in memory. Based on these results, we give a guideline how data structures should be laid out in memory to reach optimal throughput in different use cases and on different CPU architectures.

1. INTRODUCTION AND BACKGROUND

Wired networking standards and equipment continuously reach higher link speeds, with 100 Gbit/s being the current state of the art in data center networks. Packet analyzers hence need to be able to quickly scan through millions of packets and gigabytes of data to run analyses. We take a look at the performance of packet analyzers running on a stream of packets that is stored in memory. We evaluate if a carefully chosen layout can have an impact on packet analyzer performance by reducing memory access times and which values should be chosen to achieve high throughput.

Packet analyzers commonly use the pcap format [6]. Although its main purpose is on-disk storage, its layout is still relevant because the fastest and simplest way for an analyzer to read a pcap is to map the file verbatim into memory. pcap does not specify any restrictions on the alignment of packets and stores data back-to-back. Its successor pcapng [5], acknowledges this potential mistake and proposes to align all fields at 16 or 32 bit boundaries.

Naïvely aligning packets on a fixed boundary as shown in Figure 1 is not optimal because the L2 header is 14 bytes long, i.e., the L3 header is miss-aligned for 4-byte accesses such as reading IPv4 addresses. CPUs may require alignment or impose a performance penalty (up to 300% [3]) for unaligned access. Adding an additional two bytes of padding or meta-data aligns the L3 header on a four-byte boundary (see Figure 2 at the cost of two bytes of additional storage per packet. This leads to a trade-off between optimal storage and access efficiency.

We previously presented an approach to capture and analyze traffic with our tool FlowScope [4]. At its core it uses a data structure which stores packets inline together with metadata information. We chose the alignment pattern depicted in 2 based on the idea that alignment is important. However, we have not validated this performance claim until now.

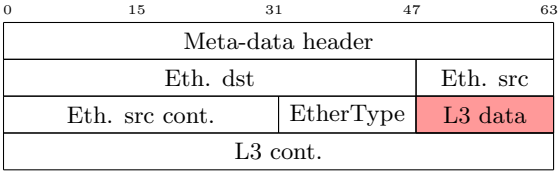


Figure 1: Naïve alignment: L3 header 2-byte aligned

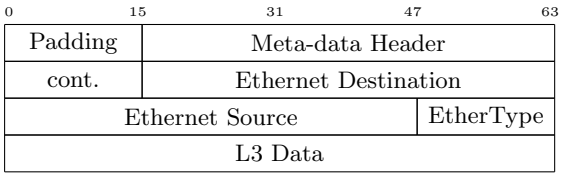


Figure 2: Padding: L3 header 4-byte aligned

CPU	Cache size	RAM channels
Intel Xeon E5-2630v4 (Q1'16)	25 MB	8 (NUMA)
Intel Xeon E5-2620v3 (Q3'14)	15 MB	4
Intel Celeron 3205U (Q1'15)	2 MB	2
Intel Atom N270 32 bit (Q2'08)	512 KB	1

Table 1: Tested CPUs

2. BENCHMARK

We develop a synthetic benchmark that generates packets and runs them through different pcap filters. Table 1 lists the tested CPUs and their properties. We generate two different packet size distributions: a uniform distribution of all sizes and a distribution based on a real passive measurement performed at the Internet uplink of the Munich Scientific Network (MWN) to the German National Research and Education Network (DFN) [2]. All header fields are filled with random values generated in a specific way to control the accept rate of the filter function. The benchmark and all code used here is available on GitHub [1].

We test both naive alignments at powers of two as well as aligned 2-byte padded packets (denoted as $X + 2$) to shift the L3 payload to an aligned address.

3. RESULTS AND ANALYSIS

Figures 3 shows the relative throughput compared to the unaligned (alignment = 1) base case. Absolute throughput was between 40 and 70 Gbit/s. Our benchmark tests several filtering functions with mostly similar results, we show a filter looking for an IP and UDP port as representative ex-

ample. The by far biggest increase in performance of 12% to 27% is achieved by aligning packets to 64 byte boundaries. This large effect occurs on all tested filters and both traffic patterns on all modern CPUs.

The interesting data in the packet is within the first 64 bytes of a packet, which could fit in a single cacheline of a modern CPU. Should the start of a packet coincidence with the start of a cache line, as it is always the case with 64 byte alignment, it is then guaranteed that any read field after the first will be served from the fast L1 cache. For the smaller alignments this is not the case: e.g., the UDP port could reside in the cache line after the EtherType, which would trigger a second expensive load from memory. It is also notable that adding an offset (64+2) does not influence the results significantly, unlike for the 4 and 8 byte boundaries. The only CPU where this is not true is the older (2008) Intel Atom CPU based on an in-order architecture which is likely limited by memory bandwidth here. We therefore conclude that on modern CPUs data locality plays a more important role than any natural alignment.

The performance benefits achieved by alignments do not come without costs. As packet sizes vary, padding bytes have to be inserted between them in the otherwise continuously used space. The storage efficiency is reduced to only 97.3% with the overhead of the 64-byte alignment based on the size distribution of our real-world traffic capture. In general, the effect is less severe with larger packets, where a few bytes padding are out-weighed by the amount of payload bytes. It is a bigger concern if truncated packets or only headers are to be stored. However, we are doing this work in the scope of FlowScope where our explicit goal is capturing the full packet data.

4. CONCLUSIONS

The benchmark results are contrary to our initial intuition and implementation in FlowScope. Performance of data aligned at cache lines increases performance by 12% to 15% with real-world data (27% with synthetic data) compared to no alignment while reducing storage efficiency by only 3%. Our benchmark is open source at GitHub [1], we are interested in performance results on other platforms.

5. REFERENCES

[1] Alignment microbenchmark. https://github.com/emmericp/align-microbenchmark.

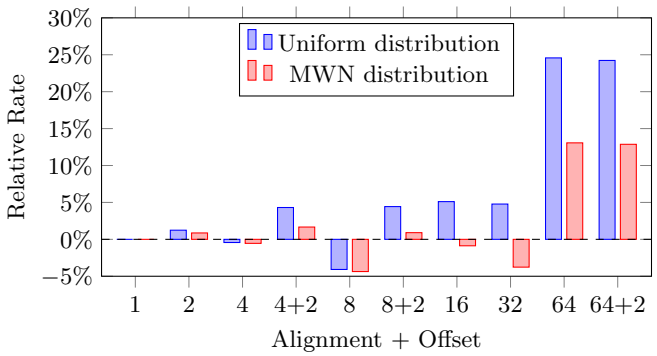
[2] Overview of the munich scientific network (mwn). https://www.lrz.de/services/netz/mwn-ueberblick_en/.

[3] U. Drepper. What every programmer should know about memory. 2007.

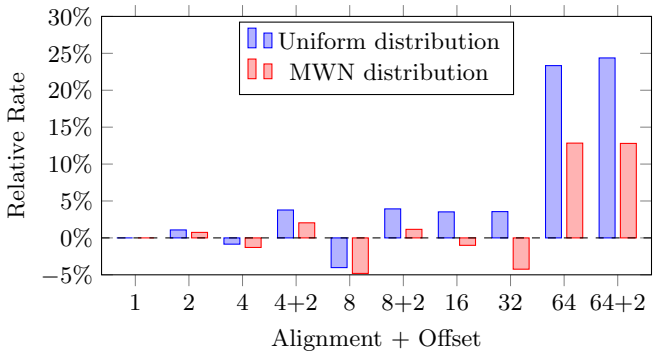
[4] P. Emmerich, M. Pudelko, S. Gallenmüller, and G. Carle. FlowScope: Efficient Packet Capture and Storage in 100 Gbit/s Networks. In *IFIP Networking 2017*, Stockholm, Sweden, June 2017.

[5] M. Tuexen, Muenster Univ. of Appl. Sciences, F. Risso, Politecnico di Torino, and J. Bongertz. PCAP Next Generation (pcapng) Capture File Format. Internet-Draft, Network Working Group, July 2017.

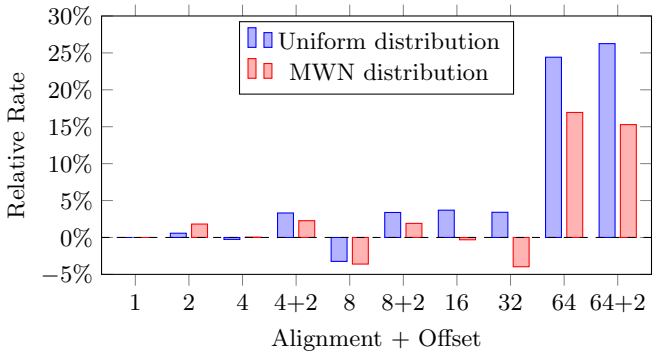
[6] Wireshark. Libpcap file format. https://wiki.wireshark.org/Development/LibpcapFileFormat.



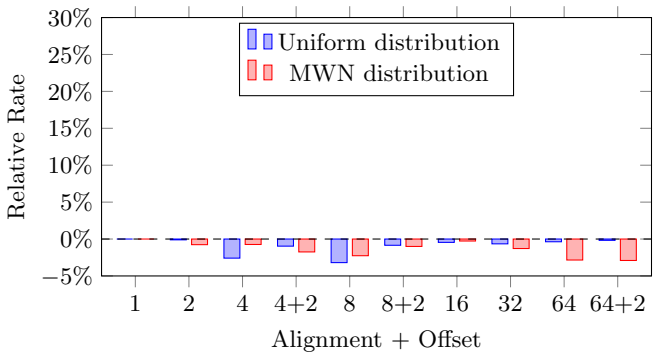
(a) E5-2630v4 CPU



(b) E5-2620v3 CPU



(c) Celeron CPU



(d) Atom CPU

Figure 3: Filtering IP address and UDP port on different CPUs