Networking Is A New Database Challenge!

Jens Helge Reelfs, Oliver Hohlfeld, Klaus Wehrle
Chair for Communication and Distributed Systems
RWTH Aachen University
{lastname}@comsys.rwth-aachen.de

ABSTRACT

Database architectures have fundamentally advanced (in-memory, parallelization, distribution) to support faster query execution and to manage higher workloads. Due to these advances, networking overheads have now become a new performance challenge, which is currently only tackled by Remote Direct Memory Access (RDMA). Fortunately, networking has recently made considerable latency and throughput improvements via kernel optimizations or by employing new architectures, e.g., kernel bypassing or specialized hardware.

We show that improved networking architectures indeed offer substantial—and currently unexplored—potential for database performance improvements regarding throughput, CPU-load, end-to-end and tail latency. To prove this potential, we used Santa, a packet processing offloading engine exemplarily implemented in the Linux kernel. Our results clearly remark reduced end-to-end latencies and throughput increases for both Memcached and MySQL by a factor of up to 1.5 and 3.3, respectively.

1. INTRODUCTION

Database systems are an important back end building block of many Internet services and thus their performance can be critical for the overall service operation. To optimize databases for speed, caching techniques, new data-structures, and new algorithms are still very active research areas. Furthermore, recent hardware advancements enable in-memory architectures which eliminate disk I/O performance bottlenecks to enable the processing of large-data volumes with short access latencies. This has lead to specialized high-performance many-core database systems that also leverage parallelism. However, further performance gains require to scale out to distributed databases, i.e., multiple machines, which introduces new challenges like data partitioning, job scheduling, concurrency, distribution [5]—and networking [1]. These efforts to provide distributed and scalable solutions can indeed yield performance-wise improved solutions. Anyhow, networking being the base component of these distributed systems also needs to improve and scale out seamlessly. The database community has identified networking as a potential bottleneck and focuses on RDMA as *the* solution by now [1], but comes at the price of requiring specific hardware and highly tailored software solutions.

At the same time, currently unexplored advances in host-level network architectures offer further potential for achieving enhanced latency and throughput figures. These advances base their performance improvements on omitting two main sources of overhead in current OS-level network stacks, caused by the kernel-user space boundary: *1)* costly context switches and system calls between kernel- and user-space and *2)* multiple memory allocations, copy operations to cross the boundary and unnecessary meta data [3]. Radical approaches try to avoid this boundary by implementing entire applications inside the kernel or by introducing specifically tailored exo-/microkernels. Another line of research aims to bypass the OS network stack entirely, either by offloading packet processing into specialized hardware, or by implementing user-level stacks with direct hardware access [2, 3]. Other approaches apply kernel optimizations or introduce new APIs, ranging from batching, pipelining to cutting overheads in buffer allocations or allow to partially offload frequent packet processing tasks into the kernel space [4]. All of these architectures offer potential to achieve higher levels of parallelization and higher throughput at lower CPU load, and lower latency.

We argue that it is now time to combine both worlds by studying the potential for database performance improvements by applying improved host-level network architectures. We exemplify this potential by evaluating two common database types, i.e., Memcached as a key-value store and MySQL as a fully-fledged database. Our evaluation bases on applying Santa as an application agnostic packet processing offloading engine which serves as an example of an improved network architecture [4]. The achieved results show latency improvements for Memcached of up to factor 1.5 and throughput increases for MySQL of up to factor 3.3.

2. IN-KERNEL PACKET PROCESSING

We base our evaluation on applying Santa [4] to both Memcached and MySQL. Santa provides an application agnostic rule processor which allows user-space applications to offload common packet processing tasks into the kernel-space, which is where the rule processor currently resides. By avoiding costly context-switches and copy operations from the kernel-space to the user-space, it offers potential for latency and throughput optimizations. It thus provides *i)* a control plane for maintaining offloaded contents and statistics, and *ii)* a data plane responsible for client facing transport.

The Santa engine takes rules comprising a *condition* determining whether or not an incoming packet matches a known request and a *processor* which is responsible for creating a well defined reply. As shown in Fig. 1, each packet destined for a Santa-enabled application is matched against all applications' installed conditions on a per-socket basis. In case of a positive match, the processor creates an answer, which is responded without further tampering the application ①.

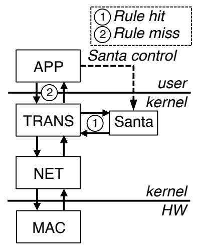
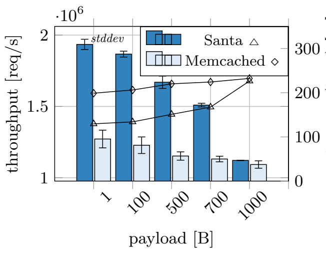
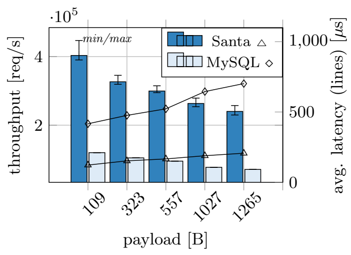


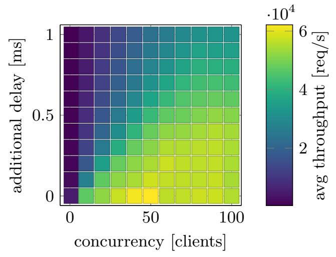
Figure 1: Santa architecture.



(a) Memcached results.



(b) MySQL results.



(c) NDB latency vs. performance.

Figure 2: Preliminary evaluation results show latency and throughput improvements.

Otherwise, the packet is forwarded to the application as usual ②. For implementing our use-cases (cf. Sec. 3), two processor methods already suffice, namely copying payload data from *j)* the incoming packet, and *jj)* a given template.

With this approach residing in the upper half of the transport layer, we cut the kernel–user space boundary and therefore costly context switches as well as memory operations.

3. EVALUATION

We present an evaluation of Santa (throughput and latency) of two use-cases: *1)* Memcached, an in-memory kv store, and *2)* MySQL, an RDBMS. Afterwards, we highlight Santa's potential within a MySQL NDB cluster setup, an in-memory scalable multinode storage engine. Our testbed setup consists of five machines each equipped with an Intel i7 4790 running at 3.6 GHz (4 cores, HT disabled), and 16 GB RAM. Networking is realized by Intel x540 10 GbE NICs using a Netgear 10 GbE switch. While varying parameters, all test series each were repeated 30 times.

Memcached, UDP transport. At first, we tested a stock Memcached followed by enabling Memcached to use Santa by offloading all keys. We customized *memaslap* to pre-populate Memcached with objects having a 16 B key and a defined value size. For the measurement, it then requests these objects in a uniform distribution.

Our observations with regard to latency and throughput are shown in Fig. 2 (a). Santa clearly decreases Memcached's `Get`-latency for small packet sizes by up to a factor of 1.5. Likewise, Santa increases the overall throughput by up to a factor of 1.5. With payloads $\geq$1 KB, Santa levels at the same latency and throughput as Memcached due to copy-operations becoming the overall predominant task.

MySQL, TCP transport. Our second use-case shows MySQL being sped up with Santa. For evaluation, we used *mysqlslap* which repeatedly selects a defined amount of data from an indexed double row (innodb) data table. We consider a fixed amount of 20 k queries requested by a varying amount of clients. For Santa, we installed rules matching the well-known requests and creating an according response.

Best results were achieved using 50 parallel clients. From our results shown in Fig. 2 (b), we observe that Santa improves performance with speedups for small payloads by up to factor 3.3. Consequently, we also observe a latency reduction of up to factor 3.4. However, the performance begins dropping at higher payloads alike seen with Memcached due to copy operations becoming predominant. Furthermore, Santa shows very little variation in throughput indicated by the min/max whiskers and clearly outperforms MySQL in all cases. Note: The observed latencies are euqally stable.

MySQL, NDB Cluster. Our additional performance evaluation of a single front and back end (to be scaled out) in-memory cluster setup highlights the benefits of applying new networking architectures to databases. We fixed the query size to 109 B while varying the concurrency and additional simulated latency (applying `tc-netem` delay on the egress path). As seen in Fig. 2 (c), the cluster naturally performs worse on this simplistic workload in comparison to a local MySQL instance due to communication overheads. These overheads become especially important at small requests challenging the network stack.

Overall, on this workload, the cluster is highly sensitive to processing & link latencies at datacenter scale, which can partly be mitigated by increasing concurrency, while remaining ultimately bounded by the host processing power.

4. CONCLUSION & FUTURE WORK

We have shown that our offloading engine Santa—as an example of new networking architectures—can increase the performance of read-dominant database applications substantially. Our approach leverages two main concepts: *1)* reducing computational expenses by offloading response packet creation and processing, and *2)* reducing networking overhead by eliminating costly context switches and copy operations. We argue that recent and currently unexplored advances in host-level network architectures have the potential to improve today's and especially tomorrow's databases, mostly independent from the workload.

We conclude that it is to be further investigated in which ways networking throughput, in-host latency and jitter affects non- and transactional distributed database solutions. Even more importantly, we must take up on the challenge of scaling recent network architectures jointly with distributed high performance database systems on many core systems.

References

[1] C. Binnig, A. Crotty, A. Galakatos, et al. The End of Slow Netw.: It's Time for a Redesign. In *VLDB*, 2016.

[2] E. Jeong and S. Wood. mTCP: a Highly Scal. User-level TCP Stack for Multicore Systems. In *NSDI*, 2014.

[3] L. Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *ATC*, 2012.

[4] F. Schmidt et al. Santa: Faster Packet Delivery for Commonly Wished Replies. In *SIGCOMM*, 2015.

[5] D. Schwalb et al. Hyrise-R: Scale-out and Hot-Standby Thr. Lazy Master Repl. for Ent. Appl. In *IMDM*, 2015.