High-Performance Main-Memory Database Systems and Modern Virtualization: Friends or Foes?

Tobias Mühlbauer* Wolf Rödiger† Andreas Kipf Alfons Kemper Thomas Neumann

Technische Universität München
{muehlbau, roediger, kipf, kemper, neumann}@in.tum.de

ABSTRACT

Virtualization owes its popularity mainly to its ability to consolidate software systems from many servers into a single server without sacrificing the desirable isolation between applications. This not only reduces the total cost of ownership, but also enables rapid deployment of complex software and application-agnostic live migration between servers for load balancing, high-availability, and fault-tolerance.

However, virtualization is no free lunch. To achieve isolation, virtualization environments need to add an additional layer of abstraction between the *bare metal* hardware and the application. This inevitably introduces a performance overhead. High-performance main-memory database systems are specifically susceptible to additional software abstractions as they are closely optimized and tuned for the underlying hardware. In this work, we analyze in detail how much overhead modern virtualization options introduce for high-performance main-memory database systems. We evaluate and compare the performance of HyPer and MonetDB under three modern virtualization environments for analytical as well as transactional workloads. Our experiments show that the overhead depends on the system and virtualization environment being used. We further show that main-memory database systems can be efficiently deployed in virtualized cloud environments such as the Google Compute Engine and that "friendship" between modern virtualization and main-memory database systems is indeed possible.

Categories and Subject Descriptors

H.2 [**Database Management**]: Systems

Keywords

Main Memory Database Systems; Virtualization

*Recipient of the Google Europe Fellowship.
†Recipient of the Oracle External Research Fellowship.
This work has further been partially sponsored by the German Federal Ministry of Education and Research (BMBF) grant RTBI 01IS12057.

DanaC'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.
ACM 978-1-4503-3724-3/15/05.
http://dx.doi.org/10.1145/2799562.2799643.

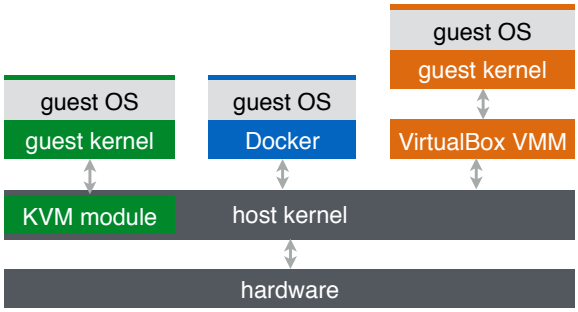


Figure 1: Modern virtualization environments

1. INTRODUCTION

Virtualization is a popular technique to isolate several virtualized environments on one physical machine and ensures that each of the environments seems to run on their own dedicated hardware resources (e.g., CPU, memory, and I/O). This enables the consolidation of software systems from many servers into a single server without sacrificing the desirable isolation between the systems. This not only reduces the total cost of ownership, but also enables rapid deployment of complex software and application-agnostic live migration between servers for load balancing, high availability, and fault-tolerance.

However, virtualization is no free lunch. The virtualization layer needs to ensure the desired isolation to avoid that software running in one virtual environment affects the stability or performance of software running in a separate environment on the same physical machine. This isolation is bound to introduce a certain overhead. In this work we analyze the impact of several modern virtualization techniques on high-performance main-memory database systems. Main-memory database systems are especially susceptible to additional software abstractions as they are often closely optimized for the underlying hardware, using advanced CPU instructions for optimal performance. They are further sensitive to memory performance as they cannot hide additional overheads behind disk access latencies. We evaluate and compare the performance of two modern main-memory database systems, HyPer [6] and MonetDB [7], running under three modern virtualization environments. We compare containerization (Docker) and hypervisors (VirtualBox and KVM+QEMU) shown in Figure 1 to bare metal performance using transactional (TPC-C, TATP) as well as analytical (TPC-H) benchmarks. As we will see, the overhead of virtualization depends heavily on the combination

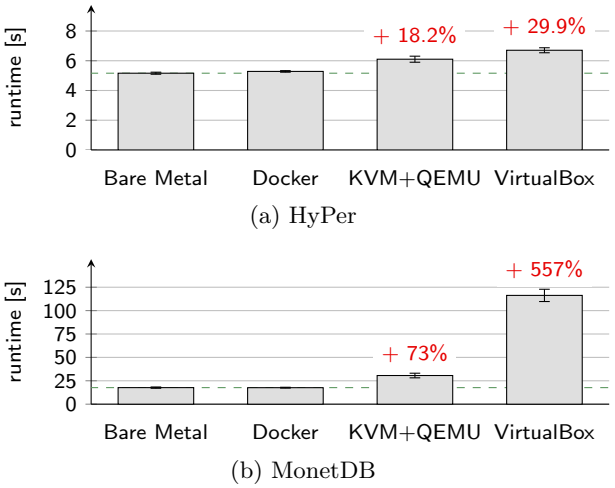


Figure 2: Running the 22 TPC-H queries (4 threads, scale factor 10, mean of 10 runs) with HyPer and MonetDB on our evaluation system (cf., Table 1)

CPU	Intel Core i7-3770
Frequency	3.40 GHz (3.90 GHz maximum turbo)
Cores/Threads	4/8
L1-/L2-Cache	32 KB/256 KB per core
L3-Cache	8 MB
Memory	32 GB DDR3 1600 MHz

Table 1: Specification of the evaluation system

of systems used, ranging from no overhead at all to severe performance degradations. We further evaluate the performance in an actual cloud environment using the Google Compute Engine that internally uses KVM.

Bare metal execution adds no performance overhead but also provides the least isolation between running processes. Sandboxing like in Google's Chrome browser can be considered application-level virtualization and adds little overhead and enables the application to manage shared resources. However, this form of isolation is prone to software bugs of a single application and this form of isolation might not be allowed for all use cases due to legal restrictions. Operating-system-level virtualization is provided by so-called "containers" in Linux, where resources are managed using cgroups. Docker is a popular example of a container management software and we use it as an instance for this category in our benchmarks. Other container managers include LXC and lmctfy. In general, for containers, the kernel is shared between the host and guest operating systems. Thus, guests cannot use a different kernel than the host (e.g., running a Windows guest on a Linux host is not possible). This limitation is, however, usually not a problem for data centers. Still, the shared kernel imposes a higher security risk than running separate kernels on the same hardware. Finally, hypervisors provide the strongest isolation guarantees and allow running multiple operating systems with different kernels on one physical machine. Among other things, hypervisors also need to isolate interrupts and accesses to memory. This is expensive and was initially performed by a software technique called binary translation. In recent years, CPU vendors added specialized instructions (e.g., Intel VT-x/EPT and AMD-V/RVI) in order to allow hardware-assisted virtualization. Both hypervisors used in our experiments, KVM+QEMU and VirtualBox, use these instructions. A downside of hypervisor-based virtualization is that the hypervisor needs to explicitly expose instruction set extensions to the underlying guest operating system. E.g., the most recent VirtualBox release (5.0.0 beta 1) that we use in our benchmarks is the first release of this hypervisor to enable the rather old SIMD instruction set extension SSE4.2 by default.

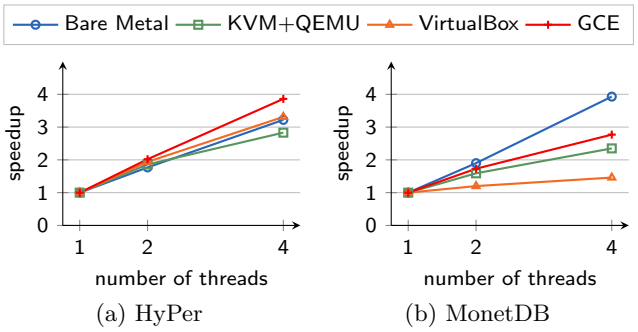


Figure 3: Scalability of TPC-H (scale factor 10) query processing in virtualized environments on our evaluation system (cf., Table 1) and on a Google Compute Engine (GCE) instance

2. BENCHMARKS

To better understand the performance of current main-memory database systems in modern virtualization environments, we benchmark our hybrid OLTP and OLAP main-memory database system HyPer [6] version 0.5-186 and MonetDB [7] version 11.19.9, a main-memory database system optimized for analytical workloads. As virtualization environments we choose the container management software Docker (version 1.5.0), the virtualization software package VirtualBox (version 5.0.0 beta 1), and the virtualization kernel module KVM (kernel 3.16.0-23) together with the QEMU hypervisor (version 2.1.0). All virtualization environments run on a Ubuntu 14.10 kernel 3.16.0-23 host operating system on an Intel Ivy Bridge CPU. Guest operating systems are also Ubuntu 14.10. For KVM+QEMU and VirtualBox, the guests are assigned 28 GB of main memory and 4 cores (virtualized cores have at least SSE 4.2). Table 1 shows the full specification of the evaluation system. In addition to the virtualized environments, we also run all benchmark workloads under the unmodified host operating system on unvirtualized hardware (bare metal). As workloads we use the analytical benchmark TPC-H (scale factor 10) and the transactional benchmarks TPC-C (5 warehouses) and TATP (1 million subscribers). Raw experimental results (as CSVs) and configuration files can be downloaded from our GitHub repository: `https://github.com/muehlbau/mmdbms-virtualized`.

Figure 2 shows the total runtime of the 22 TPC-H queries (scale factor 10) for all tested configurations when running the database systems with 4 worker threads. Runtimes are the mean of 10 runs. For HyPer, we did not measure a significant overhead compared to bare metal execution for all tested virtualization environments. Docker, as expected, added the least overhead, KVM+QEMU added

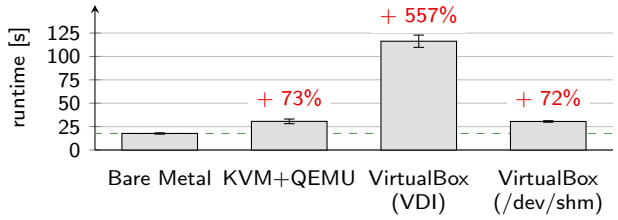


Figure 4: Storage location of database files matters when running MonetDB in VirtualBox

around 18% runtime, and VirtualBox added around 30% runtime. For MonetDB, Docker also adds almost no overhead. For KVM+QEMU and VirtualBox, on the other hand, we measured a significant overhead of 73% for KVM+QEMU and a staggering 557% for VirtualBox.

To better understand this overhead we further looked at single-threaded performance and the scalability of both database systems. Figure 3 shows the speedup with the number of worker threads for HyPer and MonetDB. We did not include Docker in this experiment, as it is almost indistinguishable from bare metal. Interestingly, when running MonetDB with only one worker thread, the overhead of KVM+QEMU and VirtualBox for MonetDB is much closer to what we measured with HyPer. It is hard to determine what exactly causes the performance degradation of MonetDB under virtualized environments—especially in VirtualBox—with more worker threads. In microbenchmarks, we were able to measure up to 10% overhead for latencies caused by TLB misses and page faults and up to 60% overhead for system calls in KVM+QEMU and VirtualBox compared to bare metal. HyPer does not suffer so heavily from these overheads as it has a different execution and parallelization model with less intermediate materialization. E.g., for TPC-H query 1, MonetDB creates a 3 GB/s write load on the memory bus (measured with Intel PCM), while HyPer only writes a few MB/s. An analysis of both database systems with strace also revealed that MonetDB issues more system calls during query execution, especially when using parallel worker threads. For VirtualBox it further depends where MonetDB's database files are stored. MonetDB maps these files into memory and this is very expensive when the backing file is stored on a VirtualBox disk image (VDI). By moving the database files to an in-memory file system in the virtual machine (/dev/shm), we were able to drastically speed up MonetDB in VirtualBox (see Figure 4).

As MonetDB is an analytical system, we measured transactional performance solely with HyPer. Figure 5 shows HyPer's TPC-C (5 warehouses) and TATP (1M subscribers) sustained transaction throughputs for the different virtualization environments. Compared to bare metal execution, virtualization adds up to 18% of overhead. Similar to the analytical benchmarks, Docker adds the least overhead, followed by KVM+QEMU, and VirtualBox.

Finally, we evaluated HyPer and MonetDB in a cloud-provisioned virtualized environment using Google Compute Engine (GCE). Internally, Google uses KVM to offer virtualized environments that can easily be provisioned on a pay-per-use basis. The instance configuration and benchmark results are shown in Table 2. Both, HyPer and MonetDB, perform very similarly compared to running in KVM+QEMU on our evaluation machine taking the lower per-core frequency (2.60 GHz compared to 3.40 GHz) and the older microarchitecture of the CPU into consideration. These are encouraging results that show that modern cloud-provisioned infrastructure and high-performance main-memory database systems can efficiently be used together.

(a) n1-standard-8 instance specification

CPU Architecture	Sandy Bridge
Frequency	2.60 GHz
Virtual Cores	8
Memory	30 GB

(b) HyPer

TPC-C (single-threaded)	88 448 transactions per second
TATP (single-threaded)	371 885 transactions per second
TPC-H (1 thread)	31.10 s (+/- 1.94 s) runtime
TPC-H (2 threads)	15.37 s (+/- 1.33 s) runtime
TPC-H (4 threads)	8.06 s (+/- 0.73 s) runtime
TPC-H (8 threads)	5.74 s (+/- 0.53 s) runtime

(c) MonetDB

TPC-H (1 thread)	104.63 s (+/- 6.76 s) runtime
TPC-H (2 threads)	60.54 s (+/- 2.16 s) runtime
TPC-H (4 threads)	37.79 s (+/- 0.77 s) runtime
TPC-H (8 threads)	35.00 s (+/- 0.82 s) runtime

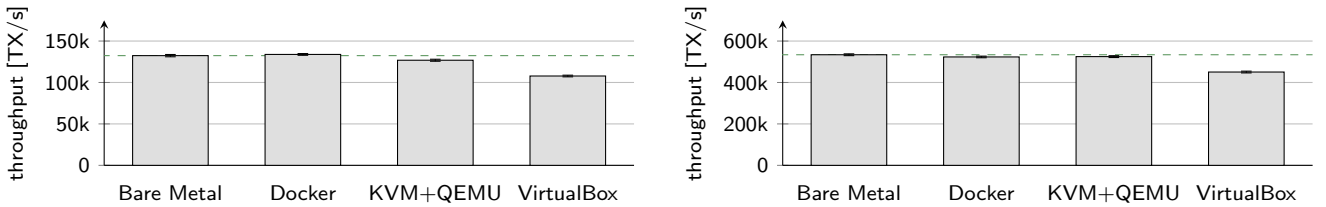
Table 2: TPC-C (5 warehouses), TATP (1M subscribers), and TPC-H queries (scale factor 10) on a n1-standard-8 Google Compute Engine instance

Of course, the question remains if resources can be used even more efficiently by consolidating multiple tenants in a single database system that "owns" the whole system. This might allow better usage of system resources compared to adding an additional layer, i.e., the virtualization layer. However, this also raises legal questions, e.g., whether sensible data of tenants can be stored together, and security concerns, e.g., whether a software bug in the database system can lead to data leaks between tenants.

3. RELATED WORK

There is only limited literature in the database field that compares the performance of database systems in virtualized environments with their native performance. Most existing work agrees that virtualization causes only a small overhead for database systems both for transactional and analytical workloads [8, 2, 5, 1, 4, 9].

Minhas et al. [8] measured the impact of virtualization on the performance of PostgreSQL in the TPC-H analytical benchmark. They found two major aspects of virtualization with Xen that can slow down performance compared to bare metal: system calls, which are up to 10× more expensive, and page faults, which take up to twice as long on virtualized hardware. The overhead for virtualized system calls does not affect the performance of PostgreSQL as system calls account only for a minor fraction of the execution time. The additional overhead for page faults on the other hand causes a significant slow down when each query is run in a separate process. Yet, using the same process for all queries reduces the overhead to only 10 % when the data is in cache and 6 % for cold caches. The cold-cache performance benefits from aggressive prefetching that hides Xen's overhead for I/O and even causes some queries to run faster than on bare metal.



(a) TPC-C (single-threaded, 5 warehouses)

(b) TATP (single-threaded, 1M subscribers)

Figure 5: Transactional benchmarks with HyPer on our evaluation system (cf., Table 1)

Curino et al. [3] promote the idea to integrate virtualization in the database system itself instead of using virtualized hardware. Their proposed Relational Cloud consists of a single database system per physical machine that manages several logical databases. They found that a single DBMS with 20 databases achieves about 6× the TPC-C throughput of 20 virtualized database system instances managing one database each. They attribute the performance degradation to multiple copies of operating/database systems and missing coordination of resources, e.g., logs and buffer pools.

The TPC-VMS [10] benchmark was developed to enable standardized comparisons for virtualized environments and adapts the TPC-C, TPC-E, TPC-H, and TPC-DS benchmarks for this purpose. TPC-VMS requires that a database system runs one of the four benchmarks simultaneously in three virtual machines that share a physical machine. Deehr et al. [4] provide TPC-VMS results for SQL Server using VMware for the transactional TPC-E benchmark. The overhead of virtualization compared to three native SQL Server instances on the same physical machine was a mere 7 %.

Grund et al. [5] measured the impacts of the Xen virtualization technology on the analytical performance of main-memory database systems. They found that the virtualized system behaved just as the physical system, except for an increased overhead for memory address translation, resulting in a minor performance degradation of 7 % for the HYRISE in-memory database system in the TPC-H benchmark.

Salomie and Alonso [9] present the Vela system that scales off-the-shelf database systems on multi-core machines and clusters using virtualization to provide a consistent view on resources. They found that main-memory workloads behave almost the same whether they are run on virtualized hardware or bare metal, while I/O-intensive workloads lead to higher CPU utilization and thus reduce performance. They attribute the absence of larger performance differences between virtualized and non-virtualized database systems mostly to the support of modern processor for virtualization, i.e., the Intel VT-x and AMD-V extensions.

Soror et al. [11] cover the *virtualization design problem*, i.e., how to allocate resources to virtual machines running on the same physical machine. This becomes especially important when different virtual machines experience different workload characteristics (e.g., CPU- vs. I/O-intensive).

4. CONCLUSION

Virtualization reduces the total cost of ownership by enabling multi-tenancy, offers rapid deployment options for applications, and can ensure high availability, load balancing, and fault tolerance via live migrations. This comes at the cost of additional overheads for the applications running in virtualized environments. Modern virtualization options differ in the degree of isolation ensured and the overhead imposed on the applications running in the virtualization environment. We have shown that containerization incurs almost no overhead and that the performance impact of hypervisor-based virtualization depends on the system being used and its configuration. While we measured only little overhead for our HyPer main-memory database system, hypervisors can significantly impact the performance of MonetDB. Finally, we have shown that main-memory database systems can be deployed in virtualized cloud environments such as the Google Compute Engine without major performance degradations.

5. REFERENCES

[1] M. Ahrens and G. Alonso. Relational databases, virtualization, and the cloud. In *ICDE*, 2011.

[2] S. Bose, P. Mishra, P. Sethuraman, and H. R. Taheri. Benchmarking database performance in a virtual environment. In *TPCTC*, 2009.

[3] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich. Relational cloud: A database-as-a-service for the cloud. In *CIDR*, 2011.

[4] E. Deehr, W. Fang, H. R. Taheri, and H. Yun. Performance analysis of database virtualization with the TPC-VMS benchmark. In *TPCTC*, 2014.

[5] M. Grund, J. Schaffner, J. Krüger, J. Brunnert, and A. Zeier. The effects of virtualization on main memory systems. In *DaMoN*, 2010.

[6] A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, 2011.

[7] S. Manegold, M. L. Kersten, and P. A. Boncz. Database Architecture Evolution: Mammals Flourished long before Dinosaurs became Extinct. *PVLDB*, 2(2), 2009.

[8] U. F. Minhas, J. Yadav, A. Aboulnaga, and K. Salem. Database systems on virtual machines: How much do you lose? In *ICDE Workshop*, 2008.

[9] T. Salomie and G. Alonso. Scaling off-the-shelf databases with vela: An approach based on virtualization and replication. *DEBU*, 38(1), 2015.

[10] W. D. Smith and S. Sebastian. Virtualization performance insights from TPC-VMS. *Transaction Processing Performance Council*, 2013.

[11] A. A. Soror, A. Aboulnaga, and K. Salem. Database virtualization: A new frontier for database tuning and physical design. In *ICDE Workshop*, 2007.