POSTER: Clouded Comparisons - On the Impact of Virtual Machines on TCP-BBR Performance

Kathrin Elmenhorst, Nils Aschenbruck
Osnabrück University, Institute of Computer Science, Germany

Abstract

Recently, TCP performance studies [1, 3, 4, 8] have focused on BBR which estimates the available bandwidth based on packet delay – instead of strictly reacting to packet loss as a sign of congestion. In contrast to packet loss, delay is more susceptible to jitter induced by the various system components in experimental setups and real-world deployments, including virtualization. Considering that BBR deployments in the wild are often running in Cloud-based virtualized environments, we measure how the choice of virtualization affects BBR v1-3 throughput in different BBR-enabled Linux kernels. Our results reveal that when using Google's custom kernels – the only publicly available source of BBRv2/v3 – BBR performance deteriorates under virtualized scheduling conditions, reducing throughput to almost zero in VirtualBox-based VMs. Overall, our work raises questions about the robustness of BBR under certain sub-optimal scheduling conditions.

CCS Concepts

• **Networks** → **Transport protocols**; *Cloud computing*; *Network performance evaluation.*

Keywords

BBR, TCP, Virtual Machine

ACM Reference Format:
Kathrin Elmenhorst, Nils Aschenbruck. 2025. POSTER: Clouded Comparisons - On the Impact of Virtual Machines on TCP-BBR Performance. In *ACM SIGCOMM 2025 Conference (SIGCOMM Posters and Demos '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3744969.3748455

1 Introduction

The parameter space of TCP performance measurements is huge: besides network (emulation) parameters, load generation, and protocol parametrization, we notice that some parameters are often underestimated in measurement studies: the Linux kernel version and virtualization. These two parameters are especially relevant in the context of the growing adoption of BBR because a) BBRv2 and v3 are only publicly available in custom kernels [5, 6], and b) virtualization is known to impact BBR more heavily than former TCP versions [7]. Thus, it is essential to understand the performance implications of different kernels and virtual machines so that we can interpret measurement results in a meaningful way.

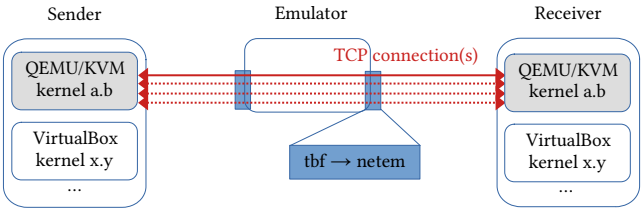


Figure 1: Measurement architecture.

BBR. BBR v1-3 are delay-based congestion control algorithms (CCAs) which aim to send at the maximum bandwidth without creating buffering latency [2]. After fast approaching the available bandwidth during the exponential slow start, BBR drains built-up queues, and varies between probing for more bandwidth (ProbeBW) and the minimum RTT (ProbeRTT) [2]. BBRv1 was added to Linux mainline kernels from version 4.19. To test newer BBR versions, Google has made two custom kernels available [5, 6] which are based on Linux v5.13 and v6.4, respectively. However, they diverge significantly in many parts of the code base, even beyond TCP.

VM hypervisor types. While type 1 hypervisors such as KVM are part of the Linux kernel and create VMs directly on the host's hardware, type 2 hypervisors, e.g., VirtualBox, run as applications on top of the host's operating system.

2 Method

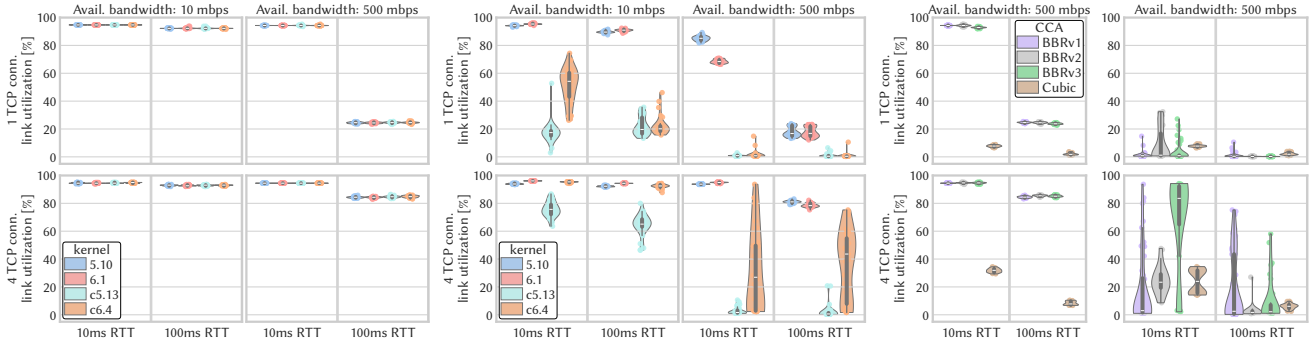
Our approach is to study TCP-BBR throughput in QEMU/KVM and VirtualBox VMs across recent mainline kernels and BBR-specific custom kernels. To this end, we consider a variety of network conditions and single vs. parallel TCP connections. As shown in Fig. 1, we perform measurements using a traditional TCP setup with two end hosts A and B, and an intermediate host as network emulator. All hosts run Debian 12 with Linux v6, and have at least 4 physical CPU cores. The network emulator forwards IP packets between A and B and employs Linux tc netem and token buckets to emulate different network conditions, i.e., uniform packet loss (0-0.1%), latency (10-100 ms RTT), and bandwidth (10-500 Mbps).

As stated above, we compare different types of VMs and different BBR-enabled Linux kernels on the end hosts. We deploy both QEMU/KVM and VirtualBox VMs which are configured with 16 GB RAM, 4 virtual CPUs, and *virtio* network adapters to connect to the host network. For the guests' Linux kernels, we install the mainline kernels used in Debian 11 and 12, i.e., v5.10 and v6.1, as well as the two custom kernels provided by Google (google/bbr) [5, 6]. We do not change kernel parameter defaults, with the exception of the maximum socket buffer sizes, which we set to the highest allowed value so that they do not become bottlenecks.

For each individual measurement, we parametrize the network emulation and start a 20-second iperf3 bulk download where we

SIGCOMM Posters and Demos '25, Coimbra, Portugal

ACM ISBN 979-8-4007-2026-0/2025/09
https://doi.org/10.1145/3744969.3748455



(a) QEMU/KVM, BBRv1 **(b) VirtualBox, BBRv1** **(c) QEMU/KVM** **(d) VirtualBox**

Figure 2: Link utilization using different VMs with 0.1% packet loss. (c-d) show google/bbr kernels only.

configure the number of parallel connections and the CCA. We repeat each run 30 times and measure the link utilization, i.e., the achieved throughput relative to the bandwidth, across all runs.

3 Results

While our evaluation covered a wider range of measurement parameters, due to space limitations, this abstract will focus on the main insights derived from our data. Fig. 2 shows the measured link utilization in the 0.1% packet loss scenario using BBRv1 on all tested kernels (2a, 2b), and additionally BBRv2, BBRv3, and Cubic on the custom kernels (2c, 2d).

QEMU/KVM. On QEMU/KVM machines (Fig. 2a, 2c), TCP can utilize almost the complete bandwidth with all BBR versions, except for the highest bandwidth delay product scenario. As a side note, when testing QEMU without KVM, we measure very similar yet slightly more variant throughput.

VirtualBox. In the setup using VirtualBox (Fig. 2b), we observe that BBR performance, especially in the custom kernels, deteriorates heavily compared to using QEMU/KVM. Specifically, custom kernel BBR drops to around 20% median link utilization for a single TCP connection in the 10 Mbps scenario, and to almost zero in the 500 Mbps scenario. With four parallel connections, custom kernel 6.4 can achieve higher aggregated throughput, but the impact in the 500 Mbps scenario is still significant. As shown in Fig. 2d, the effect occurs with all BBR versions in the custom kernels. In contrast, Cubic achieves the same results in both VMs.

For the mainline kernels, BBR performance is overall better, with most scenarios achieving the same high throughput compared to QEMU/KVM. However, in the high-bandwidth scenarios, we observe decreased throughput in v5.10 and v6.1 as well, although the reduction is less pronounced.

Investigating the root cause of this unexpected effect, we observe that the performance degradation is related to increased scheduling latency. In particular, we observe a correlation of the effect and the TCP kernel process experiencing scheduling delays in the range of multiple milliseconds. Thus, it most likely cannot achieve the goal delivery rate due to delayed sending and ACK processing. As a consequence of the non-increasing delivery rate, BBR leaves slow start prematurely and, thus, underestimate the available bandwidth. Supporting this hypothesis, TCP socket analytics (`TCP_CC_INFO`) reveal that BBR exits slow start almost immediately.

These results relate to existing work by Ha et al. [7] who theoretically analyzed degrading BBR performance during ProbeBW when multiple VMs are sharing the same host resources, and experimented using NS3 and Mininet-based setups. Indicating a potentially bigger problem, our real-system measurements show a similar effect under different scheduling conditions and during a different phase of the BBR algorithm (slow start).

Lastly, we investigate why the custom kernels are heavily affected, while the mainline kernels can remain a high throughput in most cases. When comparing the TCP source code between the google/bbr kernels and their mainline counterparts, we observe that a 2018 change in the CC module API was reverted in the custom kernels. This means that, in contrast to the mainline kernels, the custom kernels give the CC module more control of TCP burst sizes, i.e., they depend directly on BBR's current bandwidth estimate. Consequently, bandwidth underestimations have a stronger impact on the delivery rate. In contrast, patched mainline BBR keeps up burst sizes even during scheduling impairments. We successfully validated this hypothesis by adding the respective google/bbr patch #c20e56d [6] onto mainline v6.1 and repeating the measurements.

4 Takeaway and Future Work

Our findings show that BBR's throughput performance can be heavily affected by virtualized scheduling effects on the sender side, especially if BBR runs on google/bbr kernels. While the problematic scenario involving the VirtualBox hypervisor – which is not considered suitable for high performance workloads – can be seen as an edge case, our measurements raise more general questions about BBR performance under sub-optimal scheduling.

To better understand the scope and scale of the presented problem, further investigation is needed to identify and control the (virtualized) scheduling patterns triggering the observed performance degradation.

More generally, this work highlights the complexity of TCP measurement parametrization and the involved risk of overlooking relevant measurement scenarios. Thus, we aim to further investigate methods for exploring this parameter space in a more reliable and informative way.

References

[1] Yi Cao, Arpit Jain, Kriti Sharma, Aruna Balasubramanian, and Anshul Gandhi. 2019. When to use and when not to use BBR: An empirical analysis and evaluation study. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 130–136.

[2] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. https://doi.org/10.1145/3009824

[3] Soumyadeep Datta and Fraida Fund. 2023. Replication: "When to Use and When Not to Use BBR". In *Proceedings of the 2023 ACM on Internet Measurement Conference* (Montreal QC, Canada) *(IMC '23)*. Association for Computing Machinery, New York, NY, USA, 30–35.

[4] Jose Gomez, Elie F. Kfoury, Jorge Crichigno, and Gautam Srivastava. 2024. Evaluating TCP BBRv3 performance in wired broadband networks. *Computer Communications* 222 (2024), 198–208. https://www.sciencedirect.com/science/article/pii/S0140366424001658

[5] Google. 2022. Google: TCP BBR v2 Alpha/Preview Release. https://github.com/google/bbr/tree/v2alpha. [Last accessed: May 21, 2025, tag: v2alpha-2022-08-28].

[6] Google. 2024. Google: TCP BBR v3 Release. https://github.com/google/bbr/tree/v3. [Last accessed: May 21, 2025, tag: bbrv3-2024-11-22].

[7] Phuong Ha, Minh Vu, Tuan-Anh Le, and Lisong Xu. 2021. TCP BBR in Cloud Networks: Challenges, Analysis, and Solutions. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. 943–953.

[8] Danesh Zeynali, Emilia N. Weyulu, Seifeddine Fathalli, Balakrishnan Chandrasekaran, and Anja Feldmann. 2024. Promises and Potential of BBRv3. In *Passive and Active Measurement*. Springer Nature Switzerland, Cham, 249–272.