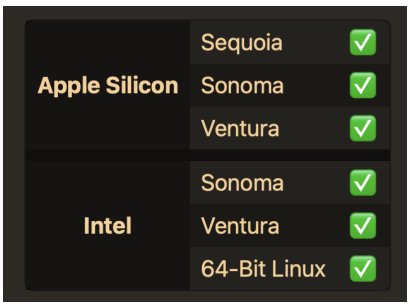
Basic Cloud Computing

Leonardo Angellotti
Università di Trieste
Trieste, Italy
leonardo.angellotti@studenti.units.it

Contents

1 Introduction

This study assesses the performance of virtual machines (VMs) [6] and containers [1] in cloud computing through standardized testing. The testing environment **not ensure a fair comparison**, in any case **is still possible to evidence some key features** needed to understand the two different approaches.
The main reason was the impossibility of using Docker on the MacbookPro (2019), having installed a more recent operating system. Here the currently binary package installation support provided by homebrew for docker [3]:



An attempt was made to use a virtual machine with Docker inside.
However, the process was uncomfortable and slow.
For this reason it was decided to use a second computer for the docker part.
In this case, however, it was possible to allocate only **one CPU per container** (the Toshiba only having two), consequently, in order to not increase the inequalities between performances, **one CPU** was assigned **per VM also** in the MacBook.

- The VM test were executed on MacBook Pro 8-Core Intel Core i9 macOS Sequoia 15.3
- The Containers test were executed on Toshiba Satellite C850-1G6 linux Mint 22.1

Two Ubuntu Desktop 16.04.7 [4] virtual machines were deployed using VirtualBox, while two Docker containers with Ubuntu:24.04 [5] were launched.

Each instance was allocated 1 CPUs and 2 GB of RAM.
The VMs (`ubuntu_1` and `ubuntu_2`) and containers (`Container_1` and `Container_2`) were connected through an internal network.
Additionally, the virtual machines were assigned 25 GB of storage to eliminate disk space as a potential performance factor, allowing CPU and memory usage to be the focus of the analysis.

Both VMs were configured with two network adapters: *Adapter 1* was set to NAT for internet access, and *Adapter 2* was configured for internal network communication between the VMs. Static IPs were assigned to the VMs (`192.168.56.100` and `192.168.56.101`) for consistency.
To facilitate remote management and testing, SSH access was enabled from the host system.

For the Docker containers, the same Docker Compose `.yalm` configuration was used. To maintain consistent CPU performance, core pinning was implemented, binding container processes to specific CPU cores and minimizing performance fluctuations. The containers communicated via an internal Docker network, providing a controlled and isolated testing environment. All the results can be checked on **Github** [2].

2 CPU Test

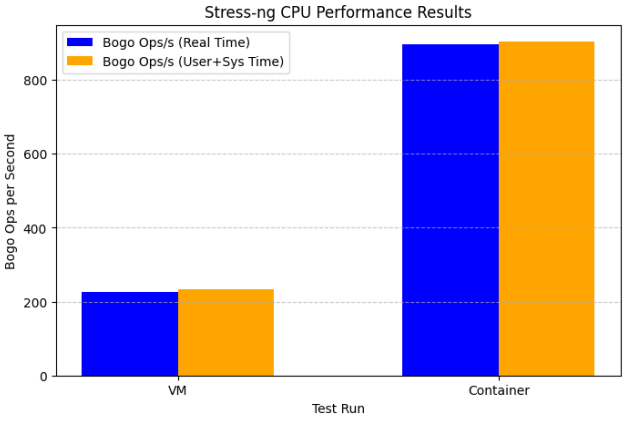
The CPU tests were made using the following command:
`stress-ng -cpu 2 -timeout 60s -metrics-brief`

The results show that containers generally outperform virtual machines (VMs) in CPU intensive workloads. This advantage stems from the **absence of a hypervisor**, which reduces resource management overhead.
For performance evaluation, I chose `bogo ops/s` as the primary metric over alternatives like `real-time`, `usr time`, and `sys time`.
Bogo Ops is a synthetic benchmark metric used to estimate

CPU performance. It does not measure actual real-world performance but provides a relative measure of computational throughput.

The term "Bogo" comes from the idea that it's an *approximation* rather than an industry-standard benchmark like SPEC or LINPACK. However, it is useful for quick comparisons between different systems or environments, such as VMs vs containers.

- `Bogo Ops/s (Real Time)`: Measures operations per second based only on the CPU time actively used, ignoring idle time.
- `Bogo Ops/s (User + Sys Time)`: Shows the number of operations completed per second based on total elapsed time, including scheduling overhead and delays.



Since containers show much higher values in both metrics compared to VMs, this suggests that containers have lower overhead and better resource efficiency than VMs.

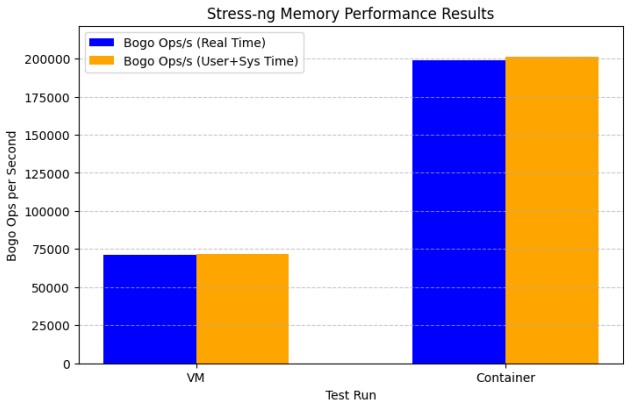
3 Memory Test

The Memory tests were made using the following command:
`stress-ng –vm 2 –vm-bytes 1G –timeout 60s –metrics-brief`

The results show that containers generally outperform virtual machines (VMs) in CPU intensive workloads.

- Containers achieve 2.5x higher memory Bogo Ops/s than VMs. This suggests lower overhead in memory operations due to the absence of a hypervisor.
- Minimal difference between "Real Time" and "User + Sys Time" for both environments. This indicates that memory operations are CPU-bound rather than waiting on external factors like I/O.
- VM performance is significantly lower, likely due to virtualization overhead.

Containers are more efficient than VMs for memory-intensive workloads. The performance improvement aligns with CPU



benchmarking trends, reinforcing that containers reduce overhead and improve resource utilization compared to VMs.

4 I/O Test

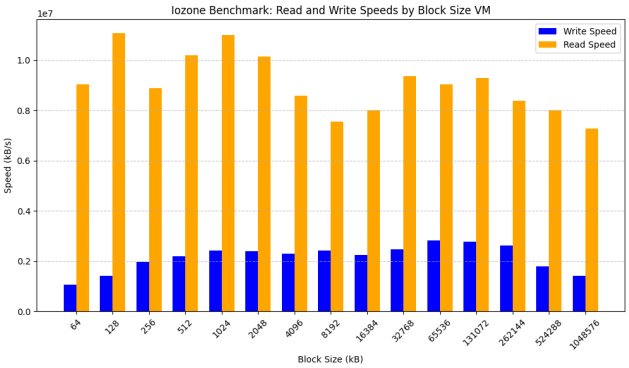
The IOzone Benchmark tests were made using the following command:

```
iozone -a -g 1G -i 0 -i 1 -i 2 -f
```

4.1 I/O Test VM

On *x-axis* are displayed a range of block sizes from `64 kB` to `1,048,576 kB` (1 GB).
On *y-axis* are reported read and write speeds in kilobytes per second (`1e7` = 10 million `kB/s`).

The disk I/O performance of virtual machines was assessed using `IOZone`. The tests measured write and read speeds across various block sizes. The results show that VMs excel in sequential read operations, with peak speeds of `1e7 kB/s` for `128 kB` block. Read operations generally outperformed write operations. However, minor fluctuations were noted at bigger block sizes, suggesting that virtualization overhead may influence I/O performance.

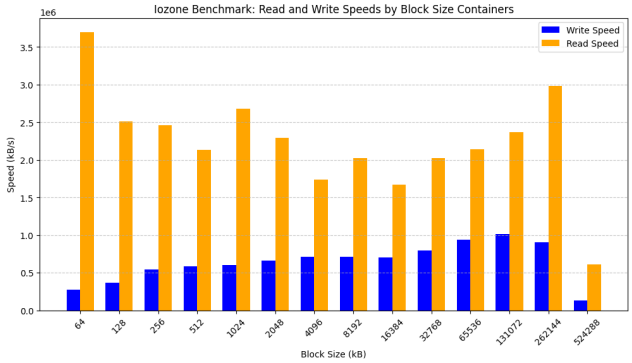


4.2 I/O Test Containers

The disk I/O performance of containers was assessed using `IOZone`. The tests measured write and read speeds across various block sizes. The results demonstrate that containers achieve high sequential read speeds, reaching up to `3.5x1e6` for `64 kB` block.

Since containers run directly on the host system's storage without the overhead of a hypervisor layer, they generally experience better I/O overhead compared to virtual machines.

In this case however, it is not possible to correctly compare the results, since it run on different machine, and capabilities for both are different. In addition, the blocks of `524288 kb` and `1048576 kb`, due to the slow calculation and inefficiency of the machine are not possible to compare.

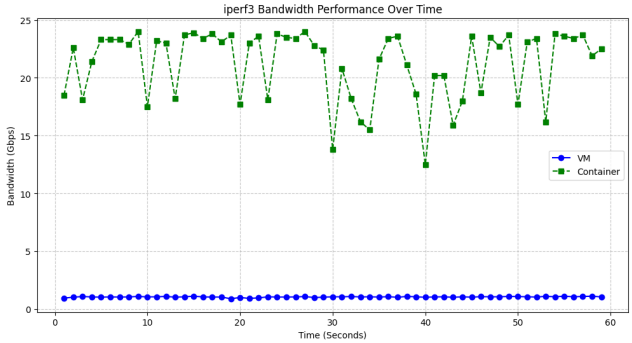


5 Network test

5.1 Single Stream VM/C

The `iperf3` Benchmark were done using the following commands:

```
iperf3 -s
iperf3 -c 192.168.56.100/101 -t 60
```

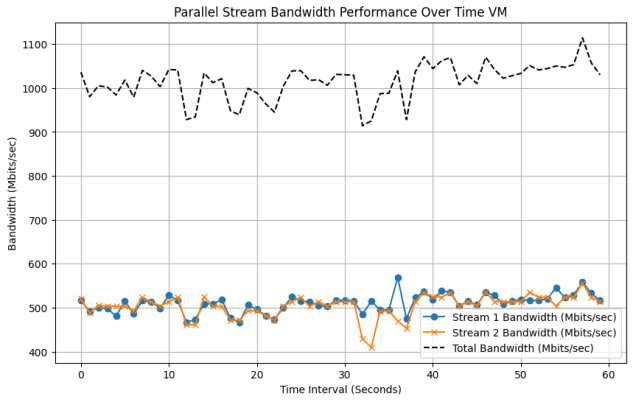


The comparative analysis highlights that containers significantly outperform VMs in network performance. Containers achieve transfer speeds over 15 times higher than VMs in single-stream tests, with bandwidth reaching 20+ `Gbps` for containers compared to just 1 `Gbps` for VMs.

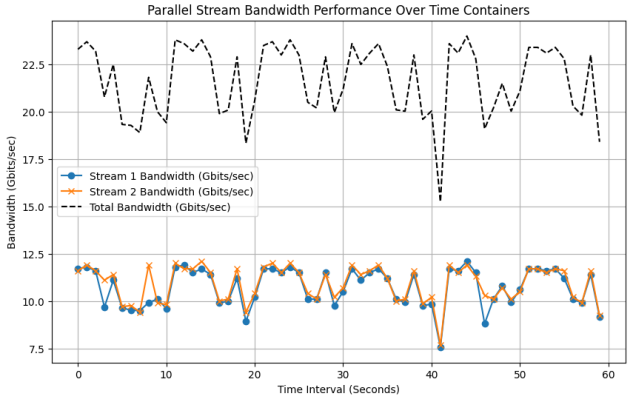
5.2 Parallel Stream

The `iperf3` for parallel Benchmark were done using the following commands:

```
iperf3 -s
iperf3 -c 192.168.56.100/101 -t 60 -P 2
```



When two parallel streams are enabled (`-P 2`), the performance gap widens further. Containers demonstrate excellent scalability, reaching up to 22 Gbps, while VMs remain limited to 1 Gbps. This stark contrast underscores how container networking benefits from direct access to the host network, bypassing the virtualization overhead of the network stack that affects VMs.



These findings highlight that while VMs offer better isolation and more structured resource management, containers deliver superior networking performance, making them the ideal choice for high-bandwidth and low-latency applications.

6 Conclusion

The test analysis revealed significant performance differences between virtual machines and containers. Containers consistently outperformed VMs across all tests, primarily due to their reduced overhead. Key observations regarding resource allocation and virtualization include:

- Virtualization latency: VMs have less efficient resource management than containers due to the presence of a hypervisor, which adds latency to CPU and memory operations.
- `I/O` operation: In Disk I/O tests, usually containers performed better in read operations, highlighting how direct resource management by the host positively impacts I/O performance, in this case however, due to the big gap between the two machine, was not possible to compare results correctly.
- Bandwidth Performance: In network tests, containers achieved significantly higher bandwidth than VMs, thanks to the absence of an extra virtualization layer for network management.

This report underscores that containers are the preferred choice for achieving maximum performance in high-intensity computing scenarios, while VMs remain a strong option for environments that require enhanced isolation and controlled resource virtualization. In any case this study has only an educational and exploratory purpose. For a better comparison it would be necessary to operate with Docker and Virtual Box on the same machine, in order to guarantee more reasonable performance comparisons.

References

[1] Docker (https://www.docker.com).
[2] Github (https://github.com/leonardoangellotti/cloud-basic).
[3] Homebrew (https://formulae.brew.sh/formula/docker).
[4] Ubuntu desktop 16.04 (https://releases.ubuntu.com/16.04/).
[5] Ubuntu server 24.04 (https://ubuntu.com/server).
[6] Virtualbox. (https://www.virtualbox.org).