Cloud Computing Report (Basic)

Stefano Lusardi

March 2025

Some details about the benchmarks can be found here `https://github.com/onafetsidrasul/Cloud`.

1 Setting up the VMs cluster

This part is based on prof. Taffoni's tutorial, so I will not go too much in detail.
The VMs cluster has been setup on VirtualBox and is based on a ***master* node** and 2 computational nodes (***node01* and *node02***). Every machine has $1024MB$ **of memory,** 1 **CPU and** $25GB$ **of disk**. The operative system I used is Ubuntu Server 24.04.2 for ARM architecture, indeed I'm working on a MacbookAir M2.
The host machine can access through ssh to the master which can access to all the nodes: this is guaranteed by the fact that the master has port forwarding and each machine is connected to an internal network named *clustervimnet*. Moreover each machine is connected to a NAT adapter for internet access.

```
vboxuser@master:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:af:bc:13 brd ff:ff:ff:ff:ff:ff
3: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:e4:59:d3 brd ff:ff:ff:ff:ff:ff
```

Figure 1: Available interfaces

The IPs are assigned dynamically thanks to the configuration of *dnsmasq* on the master node.
I also implemented a distributed filesystem using NFS at the location /shared.

2 Benchmarks

2.1 HPCC

To execute the HPCC benchmark we need to install it from apt:

```
sudo apt install hpcc
```

At this point it is necessary to configure the hpccinf.txt file, which is practically the same file as the one for hpl.
For this purpose I basically used this calculator `www.advancedclustering.com/act_kb/tune-hpl-dat-file/` setting the number of nodes = 3, cores per node = 1, block size = 300 and I also changed the size of the problem to be = 5000.
To run the benchmark I created a very simple *hostfile*

```
master slots=1
node01 slots=1
node02 slots=1
```

so I will use all the 3 machines for the benchmark. The benchmark can now be run with:

```
mpirun --hostfile hostfile -np 3 --mca btl ^vader --mca
   btl_tcp_if_include enp0s9 hpcc
```

Then I am using the 3 processors specified in the hostfile, I disabled vader (which would be useful in case I run the benchmark locally) and I specify the interface for the communication between the machines.

Note while the benchmark is running I can check that it is forwarded to all the machines with *htop*.

For all the benchmarks in the HPCC package I also tested the single machine changing the file to run everything on a single node, the rest has been kept constant.

2.1.1 HPL (floating point rate of execution for solving a linear system of equations)

	cluster	**master**	**node01**	**node02**
GFLOPS/s	70.4	47.2	47.0	47.1

While we notice that there isn't a big difference in terms of performance between the different machines we notice that there is a convenience in using the cluster instead of the single node. We'll see that this is not true in general for all the benchmarks and probably this happens because of how hpl is optimized for parallel execution.

2.1.2 DGEMM (floating point rate of execution of double precision real matrix-matrix multiplication)

	cluster	**master**	**node01**	**node02**
Avg GFLOPS/s	43.1	47.6	45.4	47.3

As before, the difference between the nodes is neglectable, but in this case there isn't a convenience in using the cluster rather than the single machine. This is probably due to the overhead introduced while the machines are communicating.

2.1.3 STREAM (sustainable memory bandwidth measurement)

Function	**cluster**	**master**	**node01**	**node02**
Copy	21.1 GB/s	68.0 GB/s	65.1 GB/s	66.2 GB/s
Scale	15.3 GB/s	49.0 GB/s	48.4 GB/s	48.0 GB/s
Add	16.3 GB/s	50.7 GB/s	51.1 GB/s	51.1 GB/s
Triad	18.7 GB/s	51.1 GB/s	51.1 GB/s	51.4 GB/s

In this case the single node overperforms the cluster probably due to Numa effect: on a cluster every node has its own ram and if a node wants to access another node's memory that is not so immediate as it is when it is accessing its own memory.

2.1.4 RandomAccess (rate of integer random updates of memory)

	cluster	**master**	**node01**	**node02**
Avg GUP/s	11.7e-2	17.8e-2	15.6e-2	18.0e-2

The worsening of the performances in the cluster with respect to the nodes can still be explained by Numa effect.

2.1.5 FFT

	cluster	**master**	**node01**	**node02**
GFLOPS/s	4.97	6.28	6.43	6.52

Ibidem.

2.2 sysbench (general system test)

To properly stress the system and get more interesting results I carried on the following benchmarks while running hpcc. The following results are obtained to test the **cpu** with the command:

```
sysbench --test=cpu --cpu-max-prime=20000 run
```

	master	node01	node02
events per second	18.7e2	18.7e2	18.7e2
avg latency (ms)	0.53	0.53	0.53

We don't see a huge difference between the different machines that indeed have the same resources and do not highlight unbalancedness between them.

The latency is small ($< 1ms$).
We can compare the results with the non-stressed scenario.

	master	node01	node02
events per second	37.9e2	37.9e2	37.6e2
avg latency (ms)	0.26	0.26	0.27

And we see that the performances are basically halved in the stressed scenario highlighting performance degradation.

We can also test the **memory** of the nodes with

```
sysbench --test=memory --memory-block-size=8K --memory-total-size=100G
   run
```

	master	node01	node02
events per second	17.1e4	17.0e4	17.1e4
avg latency (ms)	0.00	0.00	0.00

We see that we have practically no latency, so the local access to memory is not a problem, but we have seen that accessing the memory in the cluster configuration can slow down the performances of the overall cluster.

	master	node01	node02
events per second	34.0e4	34.0e4	34.0e4
avg latency (ms)	0.00	0.00	0.00

As before we notice performance degradation.

2.3 IOZone (filesystem)

These tests were run both on the single machines' filesystem and on the shared filesystem. We will concentrate on read and write scores. The command I used for the local filesystem is:

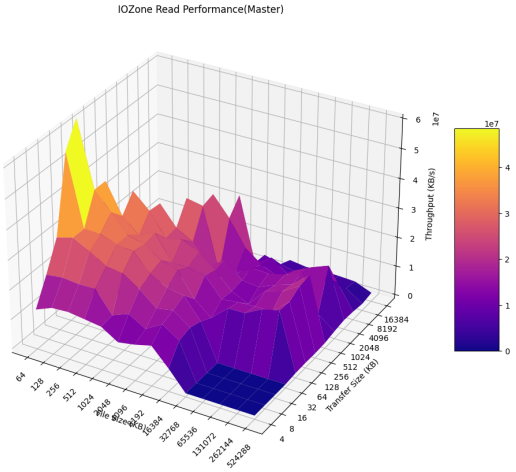
```
iozone -a -b output.xlsx
```

As in practically every test I run the performances of the different machines are comparable, so I report only the results for the master. In case you need to access also other machines' result look at the Github. Notice that there is an order of magnitude of difference in the scale of read and write graph.

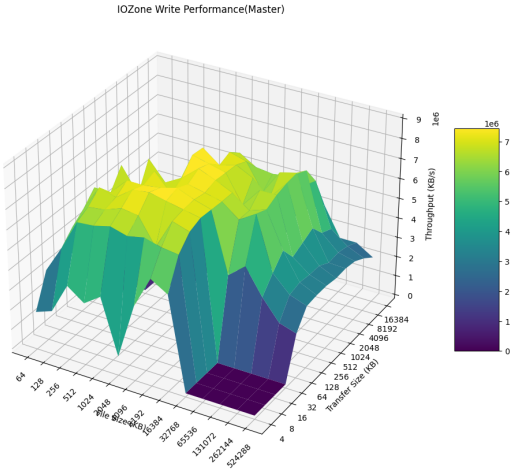
To test the distributed NFS filesystem:

```
iozone -a -b -f /shared/test
```

We don't see an appreciable difference between the local filesystem and the distributed one but we can try to see what happens to the nfs performances while stressing the network.

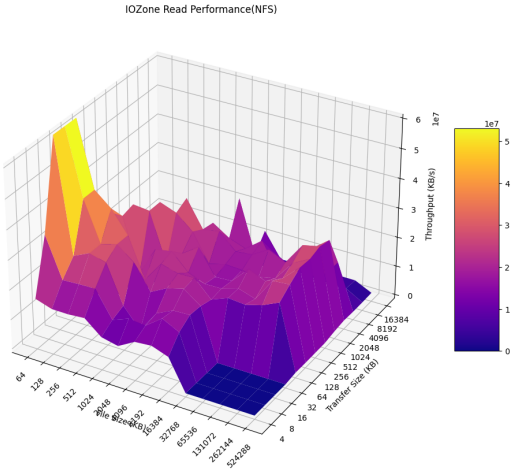


(a) Read

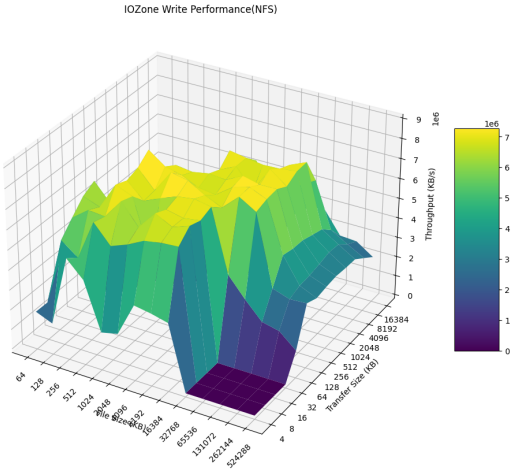


(b) Write

Figure 2: Master



(a) Read



(b) Write

Figure 3: Distributed NFS

2.4 iperf (network)

I used iperf to test the network traffic between the master and the 2 nodes. To do so I run

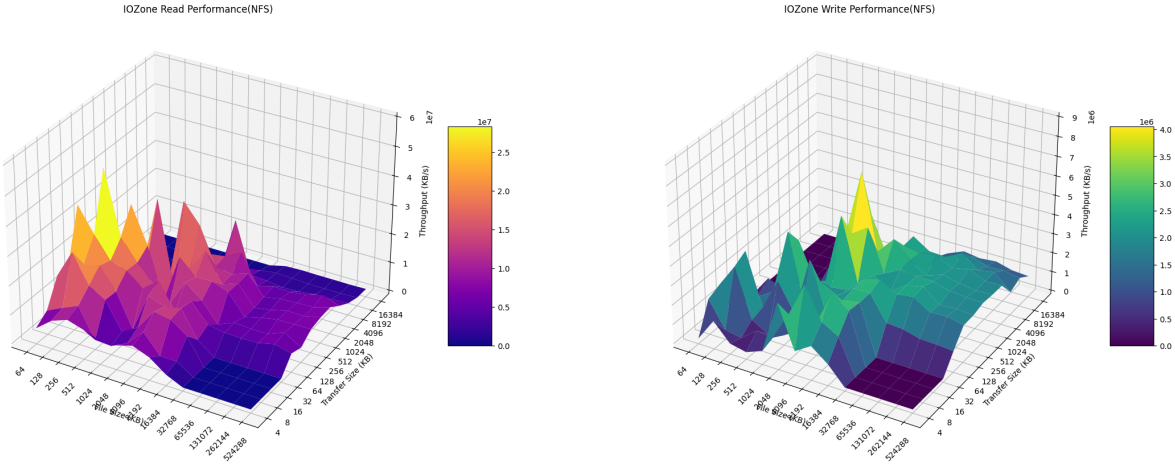
```
iperf3 -s
```

on the master and

```
iperf3 -c 192.168.0.1
```

on the nodes.
I repeated the test a lot of times due to the high variability of the results I was obtaining.
The results show a transfer speed that ranges between 3.5$Gbps$ and 4.8$Gbps$ but also a retransmission value that goes from 0 to around 10^3.
This could be a symptom of an issue related to the network, for instance, congestion, misconfiguration, or more probably latency introduced by virtualization overhead.

At this point I tried to run IOZone and iperf simultaneously setting iperf to be run for 60 seconds and rerun it in case IOZone lasts longer.
For iperf we don't see a performance degradation, while the performances of the distributed NFS filesystem shrink and become less stable.



(a) Read

(b) Write

Figure 4: Distributed NFS, stress condition

3 Docker cluster

To configure the Docker cluster I created a *dockercompose.yml* file that you can find in the repository. It basically setup a cluster of 3 machines running Ubuntu, with 1 cpu each and 1 GB of ram each. I decided to do so in order to have a similar configuration compared to the VMs one.
I also implemented core pinning.
Of course the machines are connected to an internal network, can communicate, and can access internet.

4 Benchmarks

Since the details about the different benchmarks are very similar to the ones I used for the VMs cluster, this part will be shorter and focused on a comparison between the 2 clusters.

4.1 sysbench (general system test)

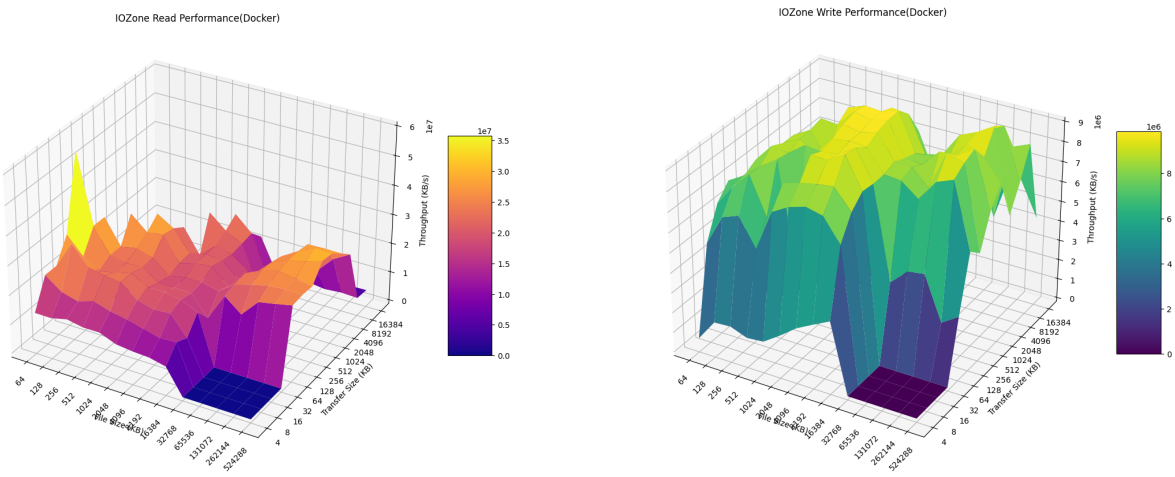
Here I took the average of the 3 machines for both the clusters.

	VMs	**Containers**
events per second (cpu)	37.8e2	39.4e2
avg latency (ms) (cpu)	0.26	0.25
events per second (memory)	34.0e4	35.7e4
avg latency (ms) (memory)	0.00	0.00

For cpu and memory we don't see a big difference between the two cases.

4.2 IOZone (filesystem)

For this benchmark I report the results of only 1 container, complete results can be found in the repository.



(a) Read

(b) Write

Figure 5: Docker

Here we can notice an improvement in the writing performance, while the reading performance is similar to the VMs one.

4.3 iperf (network)

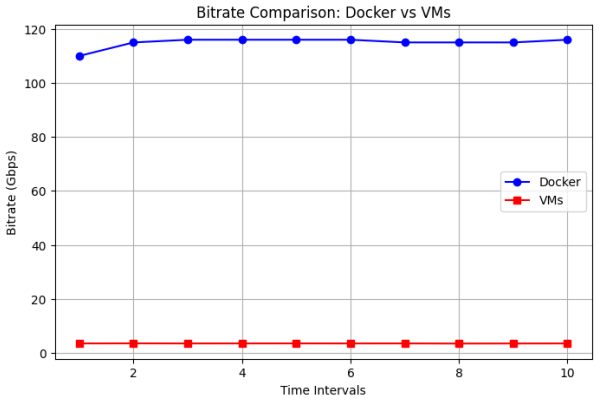


Figure 6: iperf

Even if also in this case the results in term of retransmission are very variable, the Docker cluster overperforms the transfer speed of the Vms one.
To measure latency I used ping.

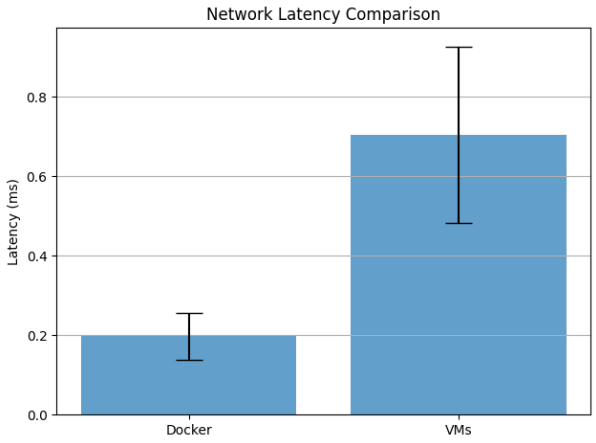


Figure 7: Latency

As we somehow now expect, the latency using the Docker cluster is very small compared to the VMs one.

5 Conclusions

Considering the overall performance of the VMs cluster, we need to take into account the poor resources allocated for it. This, together with the effect of virtualization, communication overhead and Numa effect leads to performance degradation.
Even though the Docker cluster uses the same resources we see an improvement of the overall performances due to the reduction of virtualization overhead.
Even though containers perform better than VMs, VMs can still be a reliable choice if I care more about isolation and control of resources.