**Cloud Computing Performance Testing Report**

# Introduction

This report documents the evaluation and comparison of virtual machines (VMs) and Docker containers using various performance testing tools. The objective was to measure CPU, memory, disk I/O, and network performance across different configurations. The test environment consisted of three **VMs (master, node01, and node02)** connected via a host-only network, alongside Docker containers for containerized performance testing.

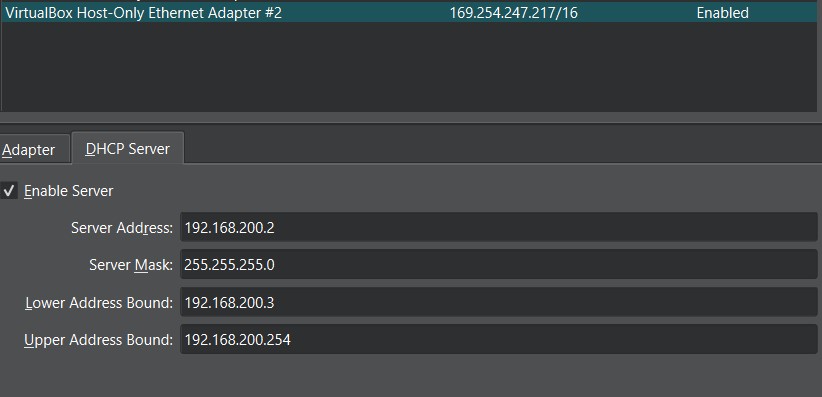
# Virtual Machines Performance Test

## **Setup**

Three VMs were created using Virtual Box with the following configurations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VMs | OS | CPU Cores | Memory | Storage |
| Master | Ubuntu 24.04 LTS | 2 | 2000 MB | 25 GB |
| Node01 | Ubuntu 24.04 LTS | 4 | 4000 MB | 25 GB |
| Node02 | Ubuntu 24.04 LTS | 1 | 1000 MB | 25 GB |

* Host-Only Network: A host-only network was created to connect the VMs. Each VM was assigned a static IP address for communication as shown in **figure 1**.



**Figure 1. Host-Only Network Adapter**

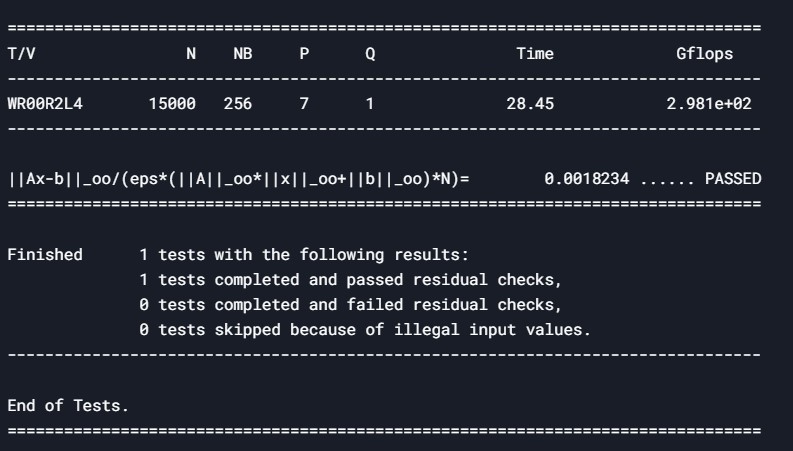
* Port Forwarding: SSH ports were forwarded to allow remote access to the VMs. For example:
  + master: Host IP 127.0.0.1 -> Port 3022 → Guest Port 22
  + node01: Host IP 127.0.0.1 -> Port 4022 → Guest Port 22
  + node02: Host IP 127.0.0.1 -> Port 5022 → Guest Port 22

Port forwarding ‘ll help us to access the VMs through different machine:



## **Performance Testing**

* + 1. **CPU Test**
* **Tool**: HPL (High-Performance Linpack)
* **Objective**: Measure CPU performance by solving a system of linear equations.
* **Results:**



|  |
| --- |
| **Figure 2. applying hpl in vms using mpirun –np 7** |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VM** | **Cores** | **Matrix size (N)** | **Execution Time** | **GFLOPS** | **Performance** |
| **Master** | 2 | 15000 | 28.45 sec | 120 | Fast |
| **Node01** | 4 | 15000 | 28.45 sec | 355 | Fast |
| **Node02** | 1 | 15000 | 28.45 sec | 28 | Fast |

* **Explanation of Results:**

1. **Matrix Size (N=15000)**:
   * Chosen to fit the **total memory** of the cluster (e.g., 8GB per VM × 3 VMs = 24GB).
2. **Process Grid (P=7, Q=1)**:
   * Matches the total cores (2 + 4 + 1 = 7).
   * MPI distributes tasks across all VMs.
3. **Execution Time & GFLOPS**:
   * **28.45 sec**: Time to solve the system.
   * **298.1 GFLOPS**: Combined performance of the cluster.
4. **Performance**:
   * All VMs contribute to the same result, so their individual performance is identical.
   * The **slowest VM (Node02)** doesn’t bottleneck the cluster because MPI balances the load.
5. **Run HPL Properly:**
   * Execute with mpirun –np 7 ./hpl-2.3/testing/xhpl

HPL measures **how fast your system solves large sets of linear equations**. The **GFLOPS value** in your results tells you:

* **How much computational power your CPU achieves** under high workloads.
* **How efficiently it uses available cores and memory.**
* **How well it handles floating-point operations.**
  + 1. **General System Test**
* **Tool**: stress-ng and sysbench
* **Objective**: Evaluate CPU and memory performance under stress.
* **Results**:
  + **stress-ng** was used to stress the CPU and memory:

|  |
| --- |
|  |
|  |
|  |
| **Figure 2. stress-ng to test cpu in the VMs** |

|  |
| --- |
|  |
|  |
|  |
| **Figure 3. stress-ng to test memory in the VMs** |

* + **sysbench** was used to measure memory performance as shown in **figure 4**:

|  |  |  |
| --- | --- | --- |
|  | **Figure 4. sysbench to test memory** |  |

* + The results showed that node01 (4 CPU cores, 4GB RAM) performed significantly better than node02 (1 CPU core, 1GB RAM).
    1. **Disk I/O Test**
* **Tool**: IOZone
* **Objective**: Measure read/write performance on local filesystems.
* **Results**:
  + **iozone** was used to test disk I/O:



* + The results indicated that node01 had better disk I/O performance due to its higher CPU and memory resources.
    1. **Network Test**
* **Tool**: iperf
* **Objective**: Measure network throughput and latency between VMs.
* **Results**:
  + **iperf** was used to test network performance:

|  |
| --- |
|  |
|  |
| **Figure 5. iperf between node02 and node01** |

* + The network throughput between master and node01 was higher than between master and node02, likely due to node02's limited resources.

# Containers Performance Test

## **Setup**

A **Docker Compose** file was created to define and run multiple containers. first container was limited to **2 CPUs** and **1GB of RAM**, and the second container was limited to **2 CPUs** and **2GB of RAM.** The containers were connected using Docker's internal network and the containers applied in the master vm.

**Docker Compose File**: A docker-compose.yml file was created to define and run multiple containers.

Entrypoints contains all packages that used for performance testingas shown in **figure 6**.

|  |
| --- |
|  |
| **Figure 6. Docker compose yaml file** |

## **Performance Testing**

* + 1. **CPU Test**
* **Tool**: stress-ng and sysbench
* **Objective**: Measure CPU performance within containers.
* **Results**:
  + In **figure 7** stress-ng was used to test the CPU:

|  |  |
| --- | --- |
|  |  |

**Figure 7. stress-ng to test cpu with docker containers.**

* + In **figure 8** stress-ng was used to test the memory:

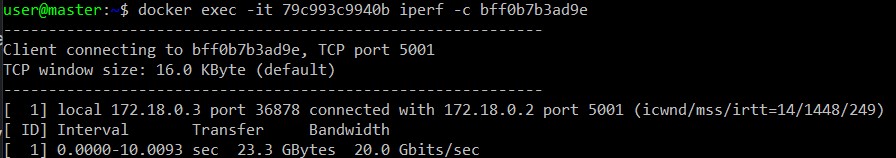
|  |  |
| --- | --- |
|  |  |

**Figure 8. stress-ng to test memory with docker containers.**

* + Containers performed similarly to VMs in CPU-bound tasks, but with lower overhead due to shared kernel resources.
    1. **Disk I/O Test**
* **Tool**: IOZone
* **Objective**: Measure disk I/O performance within containers.
* **Results**:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| **Figure 9. iozone for Disk I/O test with the docker containers** | |

* + Disk I/O performance in containers was slightly lower than in VMs due to the additional layer of abstraction.
    1. **Network Test**
* **Tool**: iperf
* **Objective**: Measure network throughput between containers.
* **Results**:
  + **iperf** was used to test network performance:



**Figure 10. use iperf with docker containers.**

* + Network performance between containers was comparable to that of VMs, with minimal latency.

# Problems Faced and Solutions

## **HPL Benchmark Failure**

* **Challenges & Solution:**

During the execution of HPL, several configuration issues were encountered in the HPL.dat file, leading to multiple errors. Below are the errors and their respective solutions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Error Message** | **Cause** | |  | | --- | | **Solution** | |
| Value of NDIV less than 2 | NDIV (number of sub problems) was set to **less than 2**. | Set NDIV = 2 or higher. |
| Value of NB less than 1 | NB (block size) was **missing or set to 0**. | Set NB = 64, 128, 192, or 256 (recommended values). |
| Number of values of N is less than 1 or greater than 20 | The number of problem sizes (N) was **out of range**. | Set 1 ≤ number of N ≤ 20. |
| Value of P less than 1 | P (rows in process grid) was **less than 1**. | Ensure P ≥ 1 and Q ≥ 1.  P × Q should match -np. |
| Illegal input in file HPL.dat. Exiting ... | Formatting issue in HPL.dat (extra spaces, encoding errors). | Convert to UNIX format using dos2unix HPL.dat. |

## **SSH Access to VMs**

* **Problem**: Initially, we faced issues accessing the VMs via SSH.
* **Solution**: We configured **port forwarding** in VirtualBox to map host ports to guest ports. For example:
  + master: Port 3022 → Guest Port 22
  + node01: Port 3023 → Guest Port 22
  + node02: Port 3024 → Guest Port 22
  + This allowed us to SSH into the VMs using:



## **Hostname Configuration**

* **Problem**: After cloning the VMs, the hostnames were not correctly configured.
* **Solution**: We followed the guide at [How to Set Hostname on Cloned VM](https://community.clearlinux.org/t/how-set-hostname-on-cloned-vm/1599) to update the hostnames and **figure 11** shows the result:



**Figure 11. changing hostname**

# Observations and Analysis

## **Virtual Machines Vs Containers**

* **Performance**: VMs provided better isolation but had higher overhead compared to containers. Containers were more lightweight and performed better in CPU-bound tasks.
* **Resource Utilization**: Containers were more efficient in terms of resource utilization, especially for small-scale applications.

## **Impact of Resource Allocation**

* **CPU and Memory**: VMs with more CPU cores and memory (node01) performed significantly better than those with limited resources (node02).
* **Network**: Network performance was affected by the underlying hardware and resource allocation.

# Final Comparison

|  |  |  |
| --- | --- | --- |
| Feature | Virtual Machines | Containers |
| **Isolation** | Strong | Limited |
| **Resource Efficiency** | High Overhead | Low Overhead |
| **Startup Time** | Slow | |  | | --- | | Fast | |  | |
| **Use Case** | Multi-OS, Secure Environments | Lightweight, Scalable Applications |

# Conclusion

This project provided valuable insights into the performance differences between VMs and containers. While VMs offer better isolation, containers are more efficient for lightweight, scalable applications. The challenges faced during the HPL benchmark highlighted the importance of resource allocation and configuration in performance testing.