Minimalist DevStack Deployment: An Analysis of Performance and Swap Utilization

Marco Ariano Kristyanto [1]*, Ahmad Miftah Fajrin [2]*
* Informatics Engineering, Universitas Surabaya (UBAYA)
marcokristyanto@staff.ubaya.ac.id [1], ahmadmiftah@staff.ubaya.ac.id [2]

Article Info

Article history:

Received 2025-06-13
Revised 2025-07-07
Accepted 2025-07-09

Keyword:

Computer Infrastructure,
Cloud Computing,
DevStack,
Grafana,
OpenStack,
Prometheus.

ABSTRACT

Cloud technology offers significant advantages; however, its high implementation costs and high hardware requirements pose barriers to small-scale deployments and educational institutions. This study addresses these challenges by investigating the performance of OpenStack deployed via DevStack on a single-node server equipped with an Intel Core i7 processor, 16 GB of RAM, and a 500 GB solid-state drive (SSD) under resource-constrained conditions. We implemented a resource tuning approach by turning off non-essential services (including Cinder, Heat, and Tempest) and adjusting Nova's memory configurations to minimize overhead. Real-time system monitoring was performed using Prometheus and Grafana to examine trends in CPU, memory, and swap utilization across three configurations: default, optimized (RAM=1024 MB), and minimalist (RAM=512 MB). Our empirical results show that the optimized setup enhances system efficiency, decreasing CPU use and memory usage from 86% to 70.90% while maintaining the ability to run up to ten virtual machines with varying operating systems (e.g., CirrOS, Ubuntu 24.04 Server LTS). However, the minimalist configurations, which aim for aggressive swap utilization and reach 100% swap saturation when running 8 VMs under idle workloads, consequently compromise overall system responsiveness despite lower CPU usage. Efficiency in this context is defined as conserving RAM and CPU usage without degrading basic system responsiveness. This highlights a critical trade-off between RAM conservation and overall system responsiveness. This research provides practical insights into designing cost-effective and lightweight OpenStack environments. It establishes a crucial threshold for memory optimization, preventing performance degradation caused by excessive swap usage, particularly in resource-constrained research settings.

I. Introduction

Cloud technology has evolved swiftly to the present time. In the era of Digital Transformation 4.0, cloud technology is rapidly proliferating across multiple domains. The use of cloud technology in education has become increasingly widespread due to the growing demand for advanced data processing. The need to access this data quickly from various locations via the internet has contributed to the increasing popularity of cloud technology, especially in educational environments.

The primary challenge institutions face in adopting cloud computing technology for learning is the high cost of commercial cloud services. Additionally, it can be difficult when custom services are required. Additionally, some lecture concepts, such as how the operating system works, require demonstration using a server or cloud.

OpenStack is an open-source cloud computing platform that provides a cloud-based operating system architecture for private and public clouds.[1] It is also used as an alternative to a private cloud, which is open-source, to create a virtual local cloud or public cloud. Due to its open-source nature, users can develop it as they wish. OpenStack also generally offers services similar to those of cloud providers today, such as Amazon Web Services (AWS) and Microsoft Azure [2].

Many studies utilize OpenStack to construct private cloud architectures that can be tailored to meet the specific needs of researchers. In-depth research was conducted by implementing OpenStack on various architectures to investigate the optimal environment for OpenStack to operate. [3]. The study results explain that OpenStack runs optimally on top of Ubuntu Server. His research on OpenStack also found that OpenStack works quite optimally in a dynamic and elastic environment, i.e., new nodes can be added dynamically. [4].

Several studies related to implementing OpenStack to optimize computer resources were also conducted. The study results found that OpenStack has the advantage of being used for small-scale cloud computing. [5] [6] [7]. OpenStack was implemented to create a private cloud, and its performance on the hardware used was monitored. The study results found that OpenStack can optimize computer hardware and resources. [8].

Several studies have investigated OpenStack performance or the implementation in educational institutions. Even fewer integrate energy efficiency as a factor in system optimization. Furthermore, the potential of DevStack as a practical instructional resource for Operating Systems and Cloud Computing courses is still inadequately examined. This research suggests and assesses a simple DevStack configuration on a small desktop server. It evaluates system performance and resource utilization through real-time monitoring tools, including Prometheus and Grafana. One critical insight revealed by this monitoring was the impact of memory optimization on swap usage. This study demonstrates that the minimal configuration (512MB per VM) can result in swap usage exceeding 100%, leading to performance degradation even under a modest workload. This confirms the impact of resource tuning on a private cloud setup.

The novelty of this study lies in discovering the performance-impacting trade-offs between RAM optimization and swap saturation, a topic that is seldom explored in lightweight DevStack installations. The combination of Prometheus and Powertop provides a comprehensive, cross-layer analysis of resource efficiency, particularly relevant for educational settings with limited infrastructure.

Unlike typical DevStack deployment studies, this work investigates explicitly how aggressive RAM allocation adjustments impact host-level swap usage in constrained single-node environments. This practical insight highlights the trade-off between resource savings and system responsiveness, a trade-off rarely discussed in small-scale OpenStack experimental contexts.

This study's contributions are:

- Implementation and assessment of a very lightweight DevStack configuration.
- Real-time system surveillance with Prometheus and Grafana.
- Evaluation of memory optimization threshold based on swap usage behavior.

The study aims to provide practical insight into resource-efficient OpenStack development, especially for those with limited hardware capability.

II. Literature Review

A. *Cloud Infrastructure in Education*

In universities, especially those that require a lot of resources related to the operating system, it is not possible to provide server resources, so it is usually necessary to use OpenStack by optimizing the IaaS services that exist on OpenStack, as done by [9]. Then, in another study, OpenStack was implemented using a lab computer. [10]The study's results indicate that utilizing OpenStack can help optimize the utilization of computer resources on campus.

Research related to the implementation of OpenStack for the creation of a private cloud for the needs of the education world was also carried out by [11]In the research, a combination of two technologies, OpenStack and Ceph, was used for educational purposes. The results were quite flexible and could be used for students' learning purposes.

B. *Lightweight cloud deployments and resource optimization*

The use of OpenStack for light-scale clouds is carried out by [4]The research utilizes OpenStack with full service and is distributed across five nodes. The study's results found that PackStack can be a solution when an elastic cloud based on OpenStack is needed.

Similar research was also conducted by [12] That uses SDN to run deployments on OpenStack. The study's results show that OpenStack can be deployed using Software-Defined Networking (SDN), thereby minimizing the overload of computing resources. The merging of OpenStack with a Hadoop Cluster was also carried out to investigate how distributed computing, including CPU and Memory usage, affects OpenStack. It was found that OpenStack can also run well under these conditions. [13].

C. *Service Level Optimization with OpenStack*

Due to its customizable nature, many researchers utilize OpenStack for resource optimization purposes, as noted by [14]. In its research, the study employs an Ubuntu server with DevStack installed. The study's results showed that the use of DevStack did not affect the CPU and RAM usage of the host OS, indicating that it can be used to run user workloads.

[14] In his research, he used the OpenStack Cinder service to compare the use of NFS and iSCSI protocols. The study's results showed that the iSCSI protocol is more suitable for implementation on OpenStack Cinder.
OpenStack service optimization is also carried out, especially for neutron services. [15]The research found that the Neutron OpenStack service can be configured and optimized for simple configuration only, especially for small-to-medium-scale clouds.

OpenStack optimization also incorporates Kubernetes, OpenStack, and serverless computing. Where the results of the study optimize the use of GPU resources, support energy optimization, and support hybrid architectures [16].

D. Infrastructure Monitoring with Prometheus and Grafana

To monitor how OpenStack is being used, you can utilize monitoring applications such as Prometheus and Grafana. This is done by [17], by using Prometheus and Grafana to monitor OpenStack usage. The results show that the use of Prometheus and Grafana helps in monitoring OpenStack optimization.

OpenStack monitoring can also be used to monitor OpenStack services and applications. As in research by [18]The article explains that OpenStack monitoring using OpenStack Exporter, ceph-exporter, and libvirt services, at least as presented in the article, demonstrates that the OpenStack service can be monitored functionally.

E. Energy Efficiency and Green Computing in Cloud

Researchers have increasingly implemented the use of OpenStack Cloud related to energy efficiency and green computing. One of them is [19], which combines OpenStack and Kubernetes for air quality monitoring. The results of the study indicate that the model's architecture can run efficiently. Monitoring and evaluation are done by monitoring Kubernetes performance.

Research related to OpenStack optimization was also conducted by [20] In his research, he proposed the concept of ECOSTAR, a model of architectural storage objects for OpenStack Swift. Development related to energy efficiency on OpenStack is also carried out by scheduling the allocation of computing resources on OpenStack. [21] [22].

The studied literature indicates that the majority of OpenStack performance assessments concentrate on large-scale or multi-node deployments. Minimal focus is directed towards swap behavior or lightweight resource optimization on single-node servers in educational environments. This study fills that gap by systematically comparing minimum DevStack installations.

III. RESEARCH METHODOLOGY

A. Research Design

This study employs a quantitative experimental methodology to assess a lightweight, single-node DevStack setup tailored for instructional purposes. The methodology encompasses system deployment, monitoring configuration, and a practical simulation of laboratory utilization. During the test, the virtual machine used was a combination of Ubuntu Server 24.04 and Cirros OS.

In this research, the swap configuration uses the default settings. No custom swap files were specified, and the kernel vm. The swapiness parameter remained at its default value of 60. The swap size used in this research is 2 GB.

B. System Architecture

This chapter discusses the research methodology and the architectural design proposed in this study. In this study, the proposed system architecture uses an assembled server with the following specifications:

- Processor Intel Core I7
- Memory RAM 16 GB
- SSD 500 GB

The architecture also supports virtualization. The DevStack is used for OpenStack because it is a type of OpenStack private cloud that can be implemented on a single node. This study employs an experimental approach using single-node deployment via DevStack.

This study uses a KVM-type hypervisor. Two clients will try to access the OpenStack server. The architectural design used is as follows.

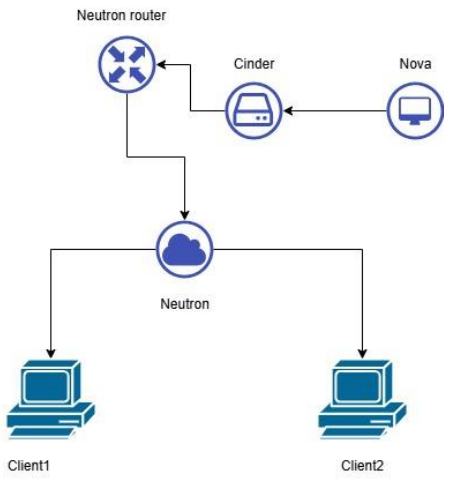


Figure 1. System Architecture

Based on Figure 1, some explanations that can be done are as follows:

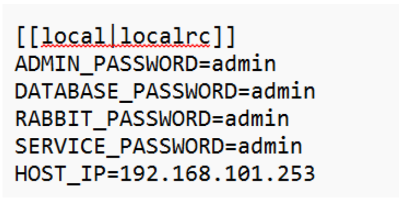
- The server is installed on the OpenStack dev stack, where all the services, namely Nova, Cinder, and Neutron, are installed.
- Then the server is connected to a MikroTik router connected to the internet.
- The client then accesses the VM instance from their PCs and laptops.

So it can be said that the components used in this study are as follows:

- a Mikrotik router for network connection to the modem and the internet
- A virtual machine that runs the Ubuntu operating system with Prometheus and Grafana, for monitoring.
- Devstack Host, which runs Ubuntu Desktop and devstack.
- Client PC for accessing the devstack.

C. *Devstack Configuration*

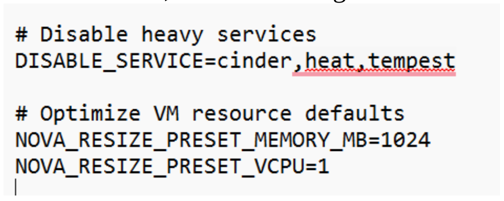
For the DevStack configuration used in this study, a comparison was conducted among three different types of configurations. Specifically, these include the general configuration (default), the optimized configuration (detuning), and the green mode configuration. Here are the various OpenStack configurations that will be tested in this study.



```
[[local|localrc]]
ADMIN_PASSWORD=admin
DATABASE_PASSWORD=admin
RABBIT_PASSWORD=admin
SERVICE_PASSWORD=admin
HOST_IP=192.168.101.253
```

Figure 2. Devstack Configuration

This research will later lead to changes in the local area. The configuration file, which contains the configuration, is shown in Figure 2. Where DevStack will be run with the default regional settings.conf configuration. Then, continue to run the devstack with a configuration that has been tuned and optimized. For the tuning version, configuration changes were made to two lines, as shown in Figure 3 below.



```
# Disable heavy services
DISABLE_SERVICE=cinder,heat,tempest

# Optimize VM resource defaults
NOVA_RESIZE_PRESET_MEMORY_MB=1024
NOVA_RESIZE_PRESET_VCPU=1
```

Figure 3. The Change in Devstack Configuration

As seen in the third picture, it is done by turning off the heat, cinder, and tempest services. Then there was an adjustment to the NOVA_RESIZE_PRESET_MEMORY_MB service, which was lowered to 1024 MB. Additionally, changes were made to the Nova service to resize the CPU preset, which was previously set to 1.

For the third configuration, namely the minimalist configuration, adjustments are made by only turning on the CLI service. Additionally, the Nova memory settings were reduced to 512 MB. All three configurations will be tested on the server and monitored to assess their performance using Prometheus and Grafana.

D. *Data Collecting and Monitoring*

For data collection, several types of parameter data will be monitored in this study. Some of these parameter data are as follows:

- CPU usage (%)
- RAM usage (%)
- The use of energy consumption is used.
- The maximum number of VMs is stable.

For improved readability in both print and digital formats, the original Grafana dashboards were color-corrected and adjusted to a light background using Photopea. This visual adjustment does not alter the underlying monitoring data.

E. *Research Methodology*

The stages of research in this study are as follows:

- Ubuntu Desktop 22.04 installation on the Server.
- Installation of Devstack, powertop, and node exporter
- Installation of Prometheus and Grafana on virtual machines to monitor the server.
- Dashboard configuration and monitoring on Prometheus and Grafana.
- VM trials are running. During testing, the VMs were booted and left idle.
- Observation and recording of performance metrics.

IV. EXPERIMENTAL RESULT

This research began by installing the Ubuntu operating system on the CPU to be tested; the Ubuntu version installed was Ubuntu 22.04. The next step is to configure the devstack according to the research scenario. In this scenario, we use the default VM swappiness value, and the swap size is 2 GB. Based on the research methodology, this study performed an OpenStack installation with the default configuration. When the installation is successful, the OpenStack dashboard is called, and the following display will appear.

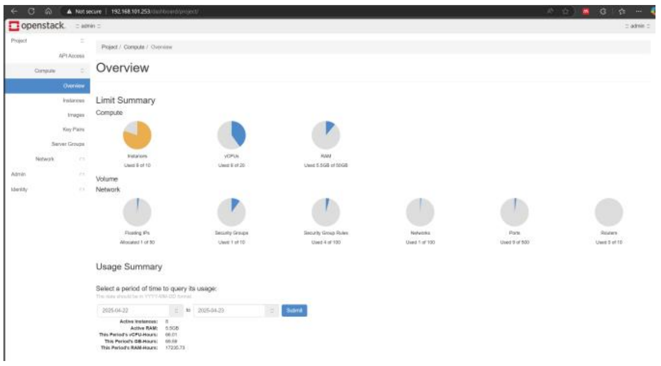


Figure 4. Dashboard Horizon

The image from Figure 4 shows the Horizon dashboard from OpenStack, which has been successfully installed. You can see how much of the maximum project quota and instances can be created. We also see a Network tab containing a list of floating IPs that can be assigned. A special virtual machine (VM) is prepared to monitor the OpenStack server, and Prometheus, Grafana, and export nodes are installed. For Prometheus and Grafana, it is installed on a virtual machine configured with a single network interface and a server.

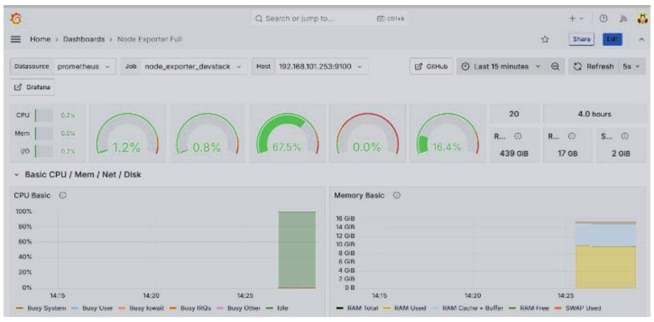
A virtual machine is prepared and connected to the network in the same subnet to monitor the performance of the server installed by OpenStack. Then, Prometheus and Grafana are installed. Monitoring is performed to observe the

DevStack and server performance, and to capture data from each configuration DevStack that runs the same 10 VMs with the same combination (6 CirrOS and 4 Ubuntu Server) using the flavor combinations listed in Table 1, with a duration of 0.5 hours. Before we delve deeper into virtual machine creation, we will explore the flavor of the OpenStack machine used in this research. It is shown in Table 1 as follows:

TABLE I.
DEVSTACK FLAVOR USED IN THIS RESEARCH

No	Machine Name	RAM – DISK – CPU Core
1.	Cirros256	256 – 1 – 1 VPCU
2.	M1.small	2048 – 20 – 1 VCPU
3.	M1.medium	4096 – 40 – 2 VCPU

This study employed several scenarios, each with different operating systems, to investigate how OpenStack performs when running instances. The scenario involves using a Cirros virtual machine and Ubuntu 24 Cloud OS. Ten virtual machines and instances are created. In the default configuration, and only for the idle host, are shown in Figure 5 below:



Home › Dashboards › Node Exporter Full
Datasource prometheus
Job node_exporter_devstack
Host 192.168.101.253:9100
Last 15 minutes
Refresh
1.2%
0.8%
67.5%
0.0%
16.4%
20
4.0 hours
439 GiB
17 GB
2 GiB
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic

Figure 5. Grafana Dashboard for the default configuration with the idle host Operating System.

The experiments continue with the creation of 10 VMs, with the combinations of 6 CirrOS and 4 Ubuntu Server 24.04. Monitoring is used with Prometheus and Grafana to investigate changes in memory usage and CPU Usage. We also use the powertop commands to monitor the CPU Usage on our host machine.

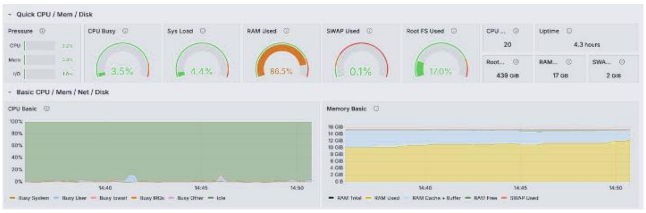
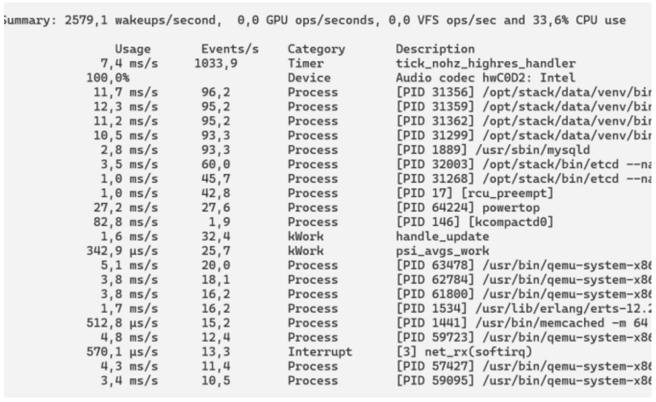


Figure 6. Grafana dashboard for the default configuration that runs 10 VMs in idle conditions



```
Summary: 2579,1 wakeups/second,  0,0 GPU ops/seconds, 0,0 VFS ops/sec and 33,6% CPU use

            Usage       Events/s    Category       Description
         7,4 ms/s     1033,9      Timer          tick_nohz_highres_handler
       100,0%                     Device         Audio codec hwC0D2: Intel
        11,7 ms/s       96,2      Process        [PID 31356] /opt/stack/data/venv/bi
        12,3 ms/s       95,2      Process        [PID 31359] /opt/stack/data/venv/bi
        11,2 ms/s       95,2      Process        [PID 31362] /opt/stack/data/venv/bi
        10,5 ms/s       93,3      Process        [PID 31299] /opt/stack/data/venv/bi
         2,8 ms/s       93,3      Process        [PID 1889] /usr/sbin/mysqld
         3,5 ms/s       60,0      Process        [PID 32003] /opt/stack/bin/etcd --na
         1,0 ms/s       45,7      Process        [PID 31268] /opt/stack/bin/etcd --na
         1,0 ms/s       42,8      Process        [PID 17] [rcu_preempt]
        27,2 ms/s       27,6      Process        [PID 64224] powertop
        82,8 ms/s        1,9      Process        [PID 146] [kcompactd0]
         1,6 ms/s       32,4      kWork          handle_update
       342,9 µs/s       25,7      kWork          psi_avgs_work
         5,1 ms/s       20,0      Process        [PID 63478] /usr/bin/qemu-system-x8
         3,8 ms/s       18,1      Process        [PID 62784] /usr/bin/qemu-system-x8
         3,8 ms/s       16,2      Process        [PID 61800] /usr/bin/qemu-system-x8
         1,7 ms/s       16,2      Process        [PID 1534] /usr/lib/erlang/erts-12.
       512,8 µs/s       15,2      Process        [PID 1441] /usr/bin/memcached -m 64
         4,8 ms/s       12,4      Process        [PID 59723] /usr/bin/qemu-system-x8
       570,1 µs/s       13,3      Interrupt      [3] net_rx(softirq)
         4,3 ms/s       11,4      Process        [PID 57427] /usr/bin/qemu-system-x8
         3,4 ms/s       10,5      Process        [PID 59095] /usr/bin/qemu-system-x8
```

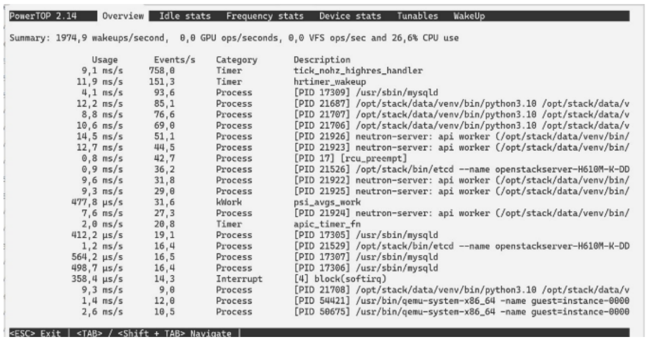
Figure 7. Powertop for the default configuration of openstack that runs 10 VMs in idle conditions

From figures 6 and 7, we can also see that memory usage increases to 86.5%, and CPU usage on powertop increases to 33.6%. We can also see that the CPU busy metric in Prometheus and Grafana remains at 3.5%, which is caused by idle conditions on each VM. The difference between CPU use in powertop and CPU busy on Prometheus and Grafana is that CPU usage metrics describe the total CPU usage on the host server, while CPU busy on Prometheus and Grafana describes the compute load performed by the CPU, where in this case, the VM being run is all in an idle condition. The results of the monitoring of the three different configurations are as follows:

TABLE II.
EXPERIMENTAL RESULT FOR THREE DEVSTACK CONFIGURATIONS

No	Metrics	Default	Optimized (memory resize =1024MB)	Minimum (memory resize=512mb)
1	Memory Usage	86.50%	70.90%	75.30%
2	Swap Usage	4.40%	30.40%	100%
3	CPU Usage	33.60%	26.60%	22.80%
5	CPU Busy	3.50%	2.10%	4.20%
6	Maximum VM run	10	10	8

A study combined measurements from the use of Powertop, Prometheus, and Grafana. The results of the study show that, for the default configuration, it can run 10 VMs with a composition of 6 CirrosOS and 4 Ubuntu servers, consuming 86.5% of memory, with the lowest swap usage at 4.4%. This indicates that the default DevStack configuration requires more resources for the idle machine VM.



```
PowerTOP 2.14   Overview   Idle stats   Frequency stats   Device stats   Tunables   WakeUp

Summary: 1974,9 wakeups/second,  0,0 GPU ops/seconds, 0,0 VFS ops/sec and 26,6% CPU use

            Usage       Events/s    Category       Description
          9,1 ms/s     758,0        Timer          tick_nohz_highres_handler
         11,9 ms/s     151,3        Timer          hrtimer_wakeup
          4,1 ms/s      93,6        Process        [PID 17309] /usr/sbin/mysqld
         12,2 ms/s      85,1        Process        [PID 21687] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/v
          8,8 ms/s      76,6        Process        [PID 21707] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/v
         10,6 ms/s      69,0        Process        [PID 21706] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/v
         14,5 ms/s      51,1        Process        [PID 21926] neutron-server: api worker (/opt/stack/data/venv/bin/
         12,7 ms/s      44,5        Process        [PID 21923] neutron-server: api worker (/opt/stack/data/venv/bin/
          0,8 ms/s      42,7        Process        [PID 17] [rcu_preempt]
          0,9 ms/s      36,2        Process        [PID 21526] /opt/stack/bin/etcd --name openstackserver-H610M-K-DD
          9,6 ms/s      31,8        Process        [PID 21922] neutron-server: api worker (/opt/stack/data/venv/bin/
          9,3 ms/s      29,0        Process        [PID 21925] neutron-server: api worker (/opt/stack/data/venv/bin/
        477,8 µs/s      31,6        kWork          psi_avgs_work
          7,6 ms/s      27,3        Process        [PID 21924] neutron-server: api worker (/opt/stack/data/venv/bin/
          2,0 ms/s      20,8        Timer          apic_timer_fn
        412,2 µs/s      19,1        Process        [PID 17305] /usr/sbin/mysqld
          1,2 ms/s      16,4        Process        [PID 21529] /opt/stack/bin/etcd --name openstackserver-H610M-K-DD
        564,2 µs/s      16,5        Process        [PID 17307] /usr/sbin/mysqld
        498,7 µs/s      16,4        Process        [PID 17306] /usr/sbin/mysqld
        358,4 µs/s      14,3        Interrupt      [4] block(softirq)
          9,3 ms/s       9,0        Process        [PID 21708] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/v
          1,4 ms/s      12,0        Process        [PID 54421] /usr/bin/qemu-system-x86_64 -name guest=instance-0000
          2,6 ms/s      10,5        Process        [PID 50675] /usr/bin/qemu-system-x86_64 -name guest=instance-0000

<ESC> Exit | <TAB> / <Shift + TAB> Navigate |
```

Figure 8. CPU Usage for Optimized (RAM=1024 MB) Configurations

In the second configuration, we can reduce RAM consumption by disabling some services, such as Heat, Cinder, and Tempest. CPU consumption is also affected by this action, reducing it to 26.6%. There was a significant decrease in memory usage to 70.9%, as described in Figure 9.

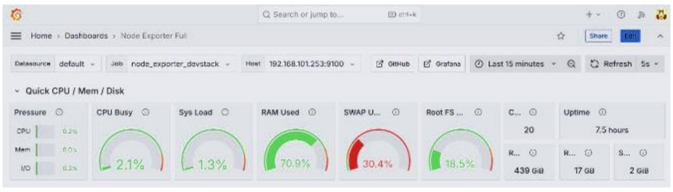
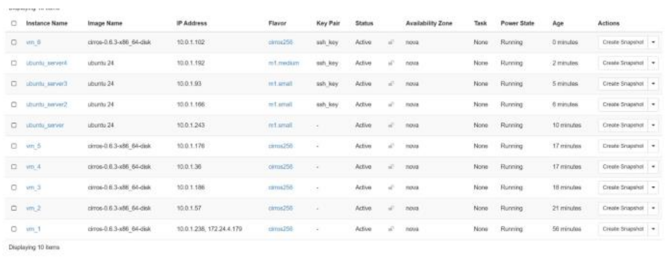


Figure 9. Grafana dashboard for the Optimized Configurations

There is also a significant improvement in swap usage, as shown in Figure 9. The increase in swap usage to 30.4% from 4.4% is caused by the reduction of Nova instances to 1024 MB and the kernel VM. swapiness parameters at its default value of 60. These adjustments, while reducing the overall RAM consumption, create increased memory pressure in the host system. Consequently, the kernel tends to proactively move less frequently used memory pages to swap pages, freeing up physical RAM, even when the VM is in an idle state. This represents an initial trade-off where the systems begin to rely more on swaps as a mechanism for RAM conservation. For a view of the number of VMs running and their specifications, see Figure 6.



Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
vm_6	cirros-0.6.3-x86_64-disk	10.0.1.102	cirros256	ssh_key	Active	nova	None	Running	0 minutes	Create Snapshot
ubuntu_server4	ubuntu 24	10.0.1.192	m1.medium	ssh_key	Active	nova	None	Running	2 minutes	Create Snapshot
ubuntu_server3	ubuntu 24	10.0.1.93	m1.small	ssh_key	Active	nova	None	Running	5 minutes	Create Snapshot
ubuntu_server2	ubuntu 24	10.0.1.166	m1.small	ssh_key	Active	nova	None	Running	6 minutes	Create Snapshot
ubuntu_server	ubuntu 24	10.0.1.243	m1.small	-	Active	nova	None	Running	10 minutes	Create Snapshot
vm_5	cirros-0.6.3-x86_64-disk	10.0.1.176	cirros256	-	Active	nova	None	Running	17 minutes	Create Snapshot
vm_4	cirros-0.6.3-x86_64-disk	10.0.1.36	cirros256	-	Active	nova	None	Running	17 minutes	Create Snapshot
vm_3	cirros-0.6.3-x86_64-disk	10.0.1.186	cirros256	-	Active	nova	None	Running	18 minutes	Create Snapshot
vm_2	cirros-0.6.3-x86_64-disk	10.0.1.57	cirros256	-	Active	nova	None	Running	21 minutes	Create Snapshot
vm_1	cirros-0.6.3-x86_64-disk	10.0.1.238, 172.24.4.179	cirros256	-	Active	nova	None	Running	56 minutes	Create Snapshot

Displaying 10 items

Figure 10. Total VMs that are running in this Devstack

The most considerable swap increase occurs in the most minimalist configuration, which sets the Nova_resize_memory amount to 512 MB. The swap memory jump becomes 100% when running 8 VMs, because using too tight Nova memory can cause poor performance on the swap memory.

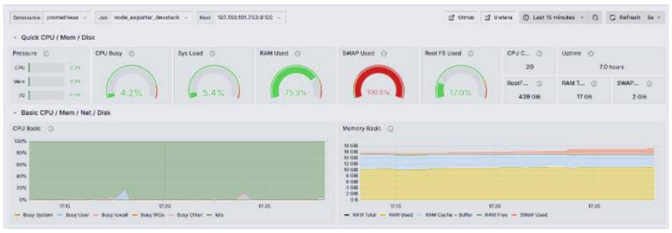
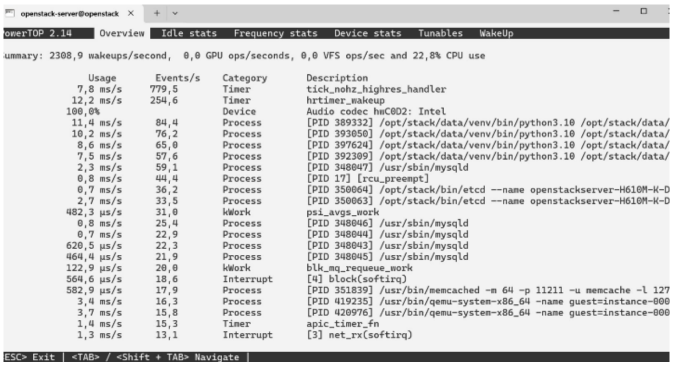


Figure 11. Grafana Dashboard for minimum configuration Devstack



```
openstack-server@openstack
owerTOP 2.14   Overview   Idle stats   Frequency stats   Device stats   Tunables   WakeUp

ummary: 2308,9 wakeups/second,  0,0 GPU ops/seconds, 0,0 VFS ops/sec and 22,8% CPU use

           Usage       Events/s    Category       Description
        7,8 ms/s     779,5        Timer          tick_nohz_highres_handler
       12,2 ms/s     254,6        Timer          hrtimer_wakeup
      100,0%                      Device         Audio codec hwC0D2: Intel
       11,4 ms/s      84,4        Process        [PID 389332] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/
       10,2 ms/s      76,2        Process        [PID 393050] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/
        8,6 ms/s      65,0        Process        [PID 397624] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/
        7,5 ms/s      57,6        Process        [PID 392309] /opt/stack/data/venv/bin/python3.10 /opt/stack/data/
        2,3 ms/s      59,1        Process        [PID 348047] /usr/sbin/mysqld
        0,8 ms/s      44,4        Process        [PID 17] [rcu_preempt]
        0,7 ms/s      36,2        Process        [PID 350064] /opt/stack/bin/etcd --name openstackserver-H610M-K-D
        2,7 ms/s      33,5        Process        [PID 350063] /opt/stack/bin/etcd --name openstackserver-H610M-K-D
      482,3 µs/s      31,0        kWork          psi_avgs_work
        0,8 ms/s      25,4        Process        [PID 348046] /usr/sbin/mysqld
        0,7 ms/s      22,9        Process        [PID 348044] /usr/sbin/mysqld
      620,5 µs/s      22,3        Process        [PID 348043] /usr/sbin/mysqld
      464,4 µs/s      21,9        Process        [PID 348045] /usr/sbin/mysqld
      122,9 µs/s      20,0        kWork          blk_mq_requeue_work
      564,6 µs/s      18,6        Interrupt      [4] block(softirq)
      582,9 µs/s      17,9        Process        [PID 351839] /usr/bin/memcached -m 64 -p 11211 -u memcache -l 127
        3,4 ms/s      16,3        Process        [PID 419235] /usr/bin/qemu-system-x86_64 -name guest=instance-000
        3,7 ms/s      15,8        Process        [PID 420976] /usr/bin/qemu-system-x86_64 -name guest=instance-000
        1,4 ms/s      15,3        Timer          apic_timer_fn
        1,3 ms/s      13,1        Interrupt      [3] net_rx(softirq)

ESC> Exit | <TAB> / <Shift + TAB> Navigate |
```

Figure 12. CPU Usage from Powertop Measurement in Minimalist Configurations (RAM = 512 MB)

Based on Figures 11 and 12, the state of the system when the minimized configurations are implemented is illustrated. As detailed in Table II, this setup allowed for running a maximum of eight stable VMs, a decrease from the default and optimized configurations. CPU usage is decreased to 22.8%, and CPU busy is at 4.20%; these numbers belie a significant underlying performance issue. The consumption of RAM usage increases to 75.30%, and a striking swap saturation occurs, reaching 100% swap usage. This complete swap saturation indicates the system is exhausting its available swap space and is under severe memory pressure, even though the VMs are running in idle workloads.

This ostensibly "low" CPU usage of 4.2% (Grafana CPU Busy) or 22.8% (Powertop) is a false measure of efficiency in this situation. Rather than carrying out calculations, it indicates that the CPU spends a significant amount of time waiting for sluggish disk I/O operations associated with swap activities. Despite the nominal CPU metrics, this phenomenon, known as "thrashing," significantly impairs overall system responsiveness and the user experience.

This configuration thus clearly shows that efficiency—which is defined as preserving RAM and CPU usage without compromising fundamental system responsiveness—was not attained. Although the raw resource consumption figures may seem lower (particularly CPU), the adverse effect of 100% swap saturation on system responsiveness, as well as the decreased capacity for stable virtual machines (from 10 to 8), points to an unfavorable trade-off. This discovery is important because it highlights a critical threshold for memory optimization: even for idle workloads, lowering virtual machine memory below a specific threshold results in an inefficient reliance on slow swap space, jeopardizing the performance that memory optimization is meant to improve. This underscores the need for more intelligent dynamic scheduling and memory balancing, which will be addressed in future multi-node implementations.

V. CONCLUSION AND FUTURE WORKS

This study demonstrates that OpenStack can be implemented on a single-node infrastructure by tuning components such as Cinder, Heat, and Tempest, as well as by adjusting the memory allocation of Nova. The results of this study showed a decrease in memory from 86% to 70.90% when running eight to ten virtual machines. However, there is an increase in swap on the OS kernel host to 100% on the most minimalist configuration (512 MB), even when idle after booting. This risk impacts the overall system performance.

However, this study has some limitations in the research methodology employed; the limitations of the network design cause difficulties in measuring more specific metrics, such as VM boot time, I/O latency, and login time. This inability is caused by the Mikrotik gateway not supporting routing and the NAT for floating IPs of the DevStack virtual machine. This limits the ability to make more specific observations of the effect of high saturation swap on virtual machine latency, and subsequently on I/O latency, which cannot be done. Additionally, since DevStack is not intended for production use, it is primarily used for experimentation and testing purposes. This finding still needs to be further tested on a larger scale.

To address the limitations of this research, the next step that can be taken is to develop it in the realm of multinode architecture. Additionally, network redesign settings are required so that the created virtual machine can be accessed from outside the DevStack network. Scheduling algorithms will be developed to avoid excessive swap usage and bottlenecks on a single node, so it is expected that these schemes and findings can be implemented at absolute scale, especially for low-cost private clouds..

REFERENCES

[1] T. Rosado and J. Bernardino, “An overview of openstack architecture,” *International Database Engineering and Applications Symposium*, 2014, doi: 10.48550/ARXIV.2210.09691.

[2] F. Mechraoui, P. Martins, and F. Caldeira, “OpenStack: a virtualisation overview,” *International Journal of Information Technology and Management*, vol. 23, no. 1, pp. 1–12, Jan. 2024, doi: 10.1504/IJITM.2024.136181.

[3] S. Pericherla, “Experimental Analysis of OpenStack Effect on Host Resources Utilization,” *Emerging Science Journal*, vol. 4, no. 6, pp. 466–492, Dec. 2019, doi: 10.28991/ESJ-2020-01246.

[4] S. Heuchert, B. P. Rimal, M. Reisslein, and Y. Wang, “Design of a small-scale and failure-resistant IaaS cloud using OpenStack,” *Applied Computing and Informatics*, vol. ahead-of-p, no. ahead-of-print, 2021, https://doi.org/10.1108/ACI-04-2021-0094.

[5] R. P. Anugrah, I. Yatini, M. A. Nugroho, and K. Kunci, “Implementasi Openstack Untuk Infrastruktur Private Cloud Computing (Studi Kasus Untuk Fasilitas Mahasiswa Utdi),” *Joism : Jurnal of Information System Management*, vol. 4, no. 1, p. 41, 2022.

[6] R. Firman, Yuhefizar, and H. Amnur, “Implementasi Openstack untuk Private Cloud pada mata kuliah Administrasi server,” *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 2, pp. 75–79, Dec. 2020, doi: 10.30630/JITSI.1.2.11.

[7] A. S. Manalu, I. M. Siregar, N. J. Panjaitan, and H. Sugara, “Rancang Bangun Infrastruktur Cloud Computing Dengan Openstack Pada Jaringan Lokal Menggunakan Virtualbox,” *Jurnal Teknik Informasi dan Komputer (Tekinkom)*, vol. 4, no. 2, p. 303, Dec. 2021, doi: 10.37600/TEKINKOM.V4I2.335.

[8] V. Jyotinagar and B. B. Meshram, “Developing an OpenStack Environment: An Exploration of Experimentation and the Diagnostic Research,” *Proceedings of the 2nd International Conference on Edge Computing and Applications, ICECAA 2023*, pp. 42–49, 2023, doi: 10.1109/ICECAA58104.2023.10212254.

[9] G. Bhatia, I. Al Noutaki, S. Al Ruzeiqi, and J. Al Maskari, “Higher Education using OpenStack,” 2018.

[10] A. Noertjahyana, H. N. Palit, and D. Kuntani, “Optimization of Computer Resources Using OpenStack Private Cloud,” *World Symposium on Software Engineering*, pp. 98–102, Sep. 2019, doi: 10.1145/3362125.3362138.

[11] L. Zhao, G. Hu, and Y. Xu, “Educational Resource Private Cloud Platform Based on OpenStack,” *Computers*, vol. 13, no. 9, Sep. 2024, https://doi.org/10.3390/computers13090241.

[12] Z. Kai, L. Youyu, L. Qi, S. C. Hao, and Z. Liping, “Building a private cloud platform based on open source software OpenStack,” in *Proceedings - 2020 International Conference on Big Data and Social Sciences, ICBDSS 2020*, Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 84–87. doi: 10.1109/ICBDSS51270.2020.00027.

[13] N. Hosamani, N. Albur, P. Yaji, M. M. Mulla, and D. G. Narayan, “Elastic Provisioning of Hadoop Clusters on OpenStack Private Cloud,” *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*, Jul. 2020, doi: 10.1109/ICCCNT49239.2020.9225277.

[14] A. Noertjahyana, H. N. Palit, R. Chandra, J. Andjarwirawan, and L. P. Dewi, “Comparative Analysis of NFS and iSCSI Protocol Performance on OpenStack Cinder Technology,” *Procedia Computer Science*, vol. 171, pp. 1498–1506, Jan. 2020, doi: 10.1016/J.PROCS.2020.04.160.

[15] P. S. S. Patchamatla, “Network Optimization in OpenStack with Neutron,” *International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*, vol. 12, no. 03, Mar. 2023, doi: 10.15662/IJAREEIE.2023.1203002.

[16] P. S. Patchamatla and I. O. Owolabi, “Integrating Serverless Computing and Kubernetes in OpenStack for Dynamic AI Workflow Optimization,” no. February, 2025, doi: 10.15680/IJMRSET.2020.0312021.

[17] L. Chen, M. Xian, and J. Liu, “Monitoring System of OpenStack Cloud Platform Based on Prometheus,” in *Proceedings - 2020 International Conference on Computer Vision, Image and Deep Learning, CVIDL 2020*, Institute of Electrical and Electronics Engineers Inc., Jul. 2020, pp. 206–209. doi: 10.1109/CVIDL51233.2020.0-100.

[18] H. Wang, X. Zhang, Z. Ma, L. Li, and J. Gao, “An Microservices-Based OpenStack Monitoring System,” *International Conference on Educational and Information Technology*, pp. 232–236, 2022, doi: 10.1109/ICEIT54416.2022.9690713.

[19] E. Kristiani, C. T. Yang, C. Y. Huang, Y. T. Wang, and P. C. Ko, “The Implementation of a Cloud-Edge Computing Architecture Using OpenStack and Kubernetes for Air Quality Monitoring Application,” *Journal on spesial topics in mobile networks and applications*, vol. 26, no. 3, pp. 1070–1092, Jun. 2020, doi: 10.1007/S11036-020-01620-5.

[20] K. A. Sanjay, S. Sivakumar, S. S. Shreya, and H. L. Phalachandra, “ECOSTAR - Energy Conservation in Swift through Tiered Storage Architecture,” *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 16–23, Nov. 2020, doi: 10.1109/CCEM50674.2020.00015.

[21] S. Alur, S. Kanamadi, S. Naik, S. Kamat, and D. G. Narayan, “Dynamic Virtual Machine Consolidation for Energy Efficiency in OpenStack-based Cloud,” *International Conference on Computing Communication and Networking Technologies*, 2023, doi: 10.1109/ICCCNT56998.2023.10307414.

[22] P. K. Prameela, P. Gadagi, R. Gudi, S. Patil, and D. G. Narayan, “Energy-Efficient VM Management in OpenStack-Based Private Cloud,” *Lecture Notes in Electrical Engineering*, vol. 735 LNEE, pp. 541–556, 2021, doi: 10.1007/978-981-33-6977-1_40.