Sprawozdanie

WSO

Migracja maszyn wirtualnych przy rozproszonej pamięci masowej

Zuzanna Godek
Mikołaj Chomański

1 Cel projektu

Celem projektu jest napisanie i przetestowanie skryptu umożliwiającego tworzenie i migrację maszyn wirtualnych. Skrypt powinien umożliwić migrację maszyn lokalnych oraz zapisanych na rozproszonej pamięci masowej, w tym także ich migrację w trakcie ich działania (live).

2 Wykorzystane technologie i urządzenia

- Python
- wirtualizator KVM
- virsh
- Network File System (NFS)
- 2 Komputery z systemem Ubuntu 22.04
- Komputer Raspberry Pi 3B+
- virt-manager do zarządzania VM

3 Przeprowadzone testy

Przetestowano tworzenie i podłączanie:

- dysku NFS
- maszyn wirtualnych

Także uruchomiono migrację:

- maszyny lokalnej z użyciem protokołu scp z wyłączeniem maszyny
- wykorzystującą NFS z wyłączeniem maszyny
- maszyny lokalnej z użyciem komendy virsh migrate live
- wykorzystująca NFS z użyciem komendy virsh migrate live

Na końcu zmierzono czas przestojów w działaniu migrowanych maszyn wirtualnych.

4 Wywołanie

Cały program to aplikacji CLI zaimplementowana w pythonie. Wywołanie polega na uruchomienie pliku main.py komendą *python3 main.py* wraz z dodatkowymi opcjami które można znaleźć używając flagi *--help*.
Większość komend skryptowych była uruchamiana z użyciem modułu subprocess.
Przykładowe wywołanie komendy widnieje poniżej:

```
result = subprocess.run(command, shell=True, capture_output=True, text=True)
```

Najważniejsze komendy (te odpowiedzialne za migrowanie) znajdują się w pliku *migrator/vm_runner.py*, natomiast funkcje odpowiedzialne za tworzenie i usuwanie maszyn znajdują się w pliku *vm_manager.py.*

5 Rozproszony system plików NFS

Stworzono trzy komendy służące do zarządzania systemem NFS. Mają one następującą składnię:

- create-nfs <client_ips> [host_ip] [folder] - stwórz serwer pozwalający połączyć się danym klientom, na danym hoście (jeżeli host_ip nie jest podane to ten system tworzony jest lokalnie), w danym folderze (domyślnie /mnt/nfs). Komenda nie ma stałego efektu i nfs będzie rozłączony po restarcie komputera.
- mount <host_ip> [host_folder] [local_folder] - zamontuj udział NFS z danego hosta (host_ip), z podanego folderu (host_folder, domyślnie /mnt/nfs), do lokalnego katalogu (local_folder, domyślnie /mnt/nfs).
- unmount [local_folder] - odmontuj udział NFS z lokalnego katalogu (local_folder, domyślnie /mnt/nfs).

Wszystkie te komendy mają opcję *--help*, która dokładniej wyjaśnia składnię.

Każda z komend zapisuje dane (np. adres IP serwera NFS, adresy IP klientów tego serwera, nazwę folderu NFS itp.) w pliku *config.json*. Są one przeznaczone do późniejszego wykorzystania w trakcie migracji.

Proces tworzenia dysku NFS na zewnętrznym hoście wygląda następująco:

```python
def create_nfs_remotely(
    host_ip: str, folder: str, client_ips: list[str], export_file: str = "/etc/exports"
):
    username = input("Enter SSH username: ").strip()

    print(f"Copying SSH key to {username}@{host_ip} ...")
    subprocess.run(
        f"ssh-copy-id -i ~/.ssh/id_rsa.pub {username}@{host_ip}", shell=True, check=True
    )

    with open("migrator/base_scripts/base_nfs_server_script.sh", "r") as f:
        script_content = f.read()

    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    rsa_key = paramiko.RSAKey.from_private_key_file(str(Path.home() / ".ssh/id_rsa"))
    ssh_client.connect(hostname=host_ip, username=username, pkey=rsa_key)

    ssh_client.exec_command("mkdir -p /tmp/nfs_migrator")
    header = f"""#!/bin/bash
        CLIENT_IPS=({" ".join(f'"{ip}"' for ip in client_ips)})
        SHARED_DIR={folder}
        EXPORTS_FILE={export_file}

        """
    sftp = ssh_client.open_sftp()
    with sftp.file("/tmp/nfs_migrator/nfs_start.sh", "w") as remote_file:
        remote_file.write(header + script_content)
        remote_file.chmod(0o755)
    sftp.close()

    ssh_client.exec_command("sudo bash /tmp/nfs_migrator/nfs_start.sh")
    ssh_client.close()
    save_hosts_config(server_ip=host_ip, client_ips=client_ips)
```

Wyniki działania komend zarządzających systemem plików NFS:

```
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~/semestr_8/wso/wso$ python3 main.py create-nfs 192.168.0.122 -H 192.168.0.167 -f /tmp/wso_nfs
Creating NFS share on 192.168.0.167 for clients: ['192.168.0.122']
Folder: /tmp/wso_nfs
Enter SSH username: mikic202
2025-05-31 19:58:40,516 [INFO] Connected (version 2.0, client OpenSSH_9.2p1)
2025-05-31 19:58:40,645 [INFO] Authentication (publickey) failed.
2025-05-31 19:58:40,666 [INFO] Authentication (publickey) successful!
SSH connection to 192.168.0.167 successful.
2025-05-31 19:58:40,800 [INFO] Connected (version 2.0, client OpenSSH_9.2p1)
2025-05-31 19:58:40,928 [INFO] Authentication (publickey) successful!
2025-05-31 19:58:41,094 [INFO] [chan 1] Opened sftp connection (server version 3)
2025-05-31 19:58:41,116 [INFO] [chan 1] sftp session closed.
Saved host configuration to config.json
```

```
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~/semestr_8/wso/wso$ python3 main.py mount 192.168.0.167 -hf /tmp/wso_nfs -lf ~/wso_test_mnt
Running command: ping -w 1 192.168.0.167
[sudo] password for mikic202:
Running command: sudo apt install nfs-common
Running command: mountpoint -q /home/mikic202/wso_test_mnt
Running command: sudo mount -t nfs 192.168.0.167:/tmp/wso_nfs /home/mikic202/wso_test_mnt
Saved host configuration to config.json
```

```
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~$ mountpoint wso_test_mnt/
wso_test_mnt/ is a mountpoint
```

```
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~/semestr_8/wso/wso$ python3 main.py unmount ~/wso_test_mnt/
Running command: mountpoint -q /home/mikic202/wso_test_mnt/
Unmounting /home/mikic202/wso_test_mnt/ ...
Running command: sudo umount -f -l /home/mikic202/wso_test_mnt/
/home/mikic202/wso_test_mnt/ unmounted
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~/semestr_8/wso/wso$ mountpoint ~/wso_test_mnt/
/home/mikic202/wso_test_mnt/ is not a mountpoint
(pt) (pt) mikic202@mikic202-Lenovo-Legion-5-15IMH05H:~/semestr_8/wso/wso$
```

6 Tworzenie maszyn wirtualnych

Stworzyliśmy dwie komendy służące do tworzenia i usuwania maszyn wirtualnych wykorzystujących virsh. Te komendy to:

- create-vm <img_name> <image_path> <vm_name> [disk_size] [ram_size] - utwórz maszynę wirtualną na bazie obrazu (img_name) i ścieżki do pliku lub linku (image_path), nadaj jej nazwę (vm_name), opcjonalnie określ rozmiar dysku (domyślnie: 10 GB) i RAM (domyślnie: 1024 MB).
- delete-vm <vm_name> - usuń maszynę wirtualną o podanej nazwie (vm_name).

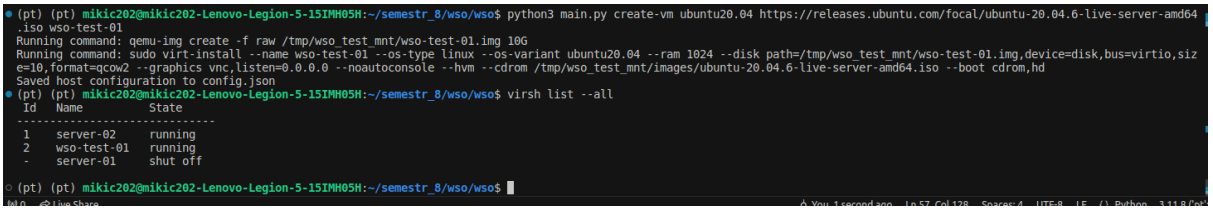
Ich implementacja polega na wywoływaniu odpowiednio komend virt-install do tworzenia, virsh destroy oraz virsh undefine. Dodatkowo w trakcie instalacji z użyciem linku do dystrybucji systemu, uruchamiana jest komenda wget pobierająca do folderu NFS (jeżeli dana dystrybucja jeszcze nie jest pobrana) wybrany obraz.
Proces tworzenia maszyny wygląda następująco:

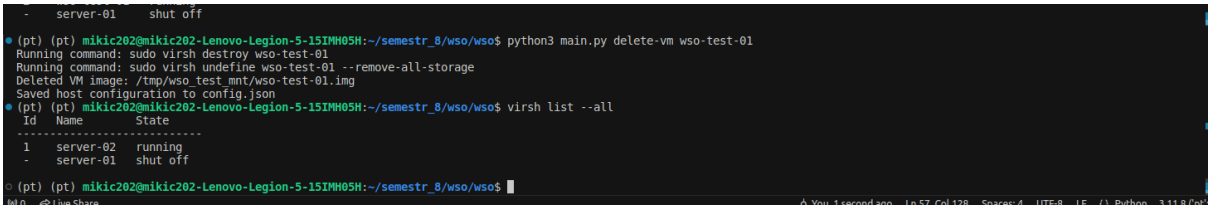
```python
def create_vm_on_nfs(
    vm_name: str, image_path: str, image_name: str, disk_size: int, ram_size: int
):
    config = read_hosts_config()
    if not config or "nfs_path" not in config or config["nfs_path"] == "":
        print("No NFS configuration found. Please run the setup command first.")
        raise typer.BadParameter()
    nfs_path = Path(config["nfs_path"])
    image_path = get_path_to_image(image_path, image_name, nfs_path)
    run_command(f"qemu-img create -f raw {nfs_path/vm_name}.img {disk_size}G")

    _create_vm(
        vm_name,
        image_name,
        nfs_path / image_path,
        disk_size,
        ram_size,
        nfs_path / f"{vm_name}.img",
    )
    save_hosts_config(vm_names=config["vm_names"] + [vm_name])
```

Najpierw znajdujemy bezwzględną ścieżkę do obrazu systemu. Jeżeli wprowadzono link do obrazu to bazując na jego nazwie sprawdzamy czy nie został zcache'owany wcześniej, jeżeli tak to znajduje jego lokalizację i ją zwraca, a jeżeli nie to używa komendy wget do pobrania pliku (poprawność linku nie jest sprawdzana). Potem tworzony jest dysk, który dana maszyna będzie wykorzystywać. Następnie uruchamiana jest komenda virsh-install wyglądającą następująco:
sudo virt-install --name {vm_name} --os-type linux --os-variant {image_name} --ram {ram_size} --disk {disk_path},device=disk,bus=virtio,size={disk_size},format=qcow2 --graphics vnc,listen=0.0.0.0 --noautoconsole --hvm --cdrom {image_path} --boot cdrom,hd
Funkcja sugeruje że maszyna tworzona jest na dysku NFS, ale ścieżka *nfs_path* to ścieżka lokalna więc może być ta komenda wykorzystana również do lokalnej instalacji.

Usuwanie maszyn to wywołanie komend
sudo virsh destroy
oraz
sudo virsh undefine {vm_name} --remove-all-storage

Wyniki komend widnieją poniżej:





7 Migracja maszyn

Stworzono 4 komendy służące do migracji maszyn. Te komendy to:

- migrate-local-live <vm_name> <host_ip> - wykonaj migrację live lokalnej maszyny wirtualnej (vm_name) z innego hosta (host_ip) do obecnego hosta.
- migrate-nfs-live <vm_name> <host_ip> - wykonaj migrację live maszyny wirtualnej (vm_name) działającej z udziałem NFS z innego hosta (host_ip) do obecnego hosta.
- migrate-nfs <img_name> - uruchom maszynę wirtualną z pliku (img_name) znajdującego się na NFS.
- migrate-scp <host_ip> <img_name> - skopiuj i uruchom maszynę wirtualną z innego hosta (host_ip) na podstawie pliku (img_name).

Dwa pierwsze polecenia wykorzystują komendę vish migrate live w celu przeniesienia działającej maszyny. Pierwsza przenosi maszynę wirtualną wraz z jej pamięcią, natomiast druga wymaga, aby maszyna wirtualna była podłączona no NFSa (ścieżki do folderów z NFSem muszą być takie same na obu hostach). Obie komendy są wykonywane przy użyciu

ssh u hosta, na którym obecnie działa maszyna, żeby przenieść ją do hosta, który wywołuje skrypt. Funkcja wykonująca migrację live z kopiowaniem dysku wygląda następująco:

```python
def migrate_live_local(vm_name: str, host_ip: str):
    log_info_before(vm_name=vm_name, host_ip=host_ip)
    if not is_host_available(host_ip):
        raise RuntimeError(f"Host {host_ip} is not reachable")

    ssh_user = input(f"Enter SSH username for {host_ip}: ").strip()
    password = getpass.getpass(f"Sudo password for {ssh_user}@{host_ip}: ")
    current_ip = get_local_ip()

    virsh_command = [
        "ssh",
        f"{ssh_user}@{host_ip}",
        "sudo",
        "-S",
        "virsh", "migrate",
        "--live", "--persistent", "--unsafe", "--verbose",
        "--copy-storage-all",
        f"{vm_name}",
        f"qemu+ssh://{ssh_user}@{current_ip}/system",
        "--migrateuri", f"tcp://{ssh_user}@{current_ip}:49153"
    ]

    virsh_result = subprocess.run(
        virsh_command,
        input=password + "\n",
        capture_output=True,
        text=True,
    )

    if virsh_result.returncode == 0:
        logging.info(f"Machine migrated successfully.\n{virsh_result}")
    else:
        raise RuntimeError(f"Failed to copy VM image:\n{virsh_result.stderr}")
```

Główną częścią jest komenda virsh migrate. Opcja --live zapewnia że maszyna będzie działać podczas migracji, --persistent zapewnia zachowanie maszyny na hoście z którego została zmigrowana. Opcje --unsafe i --verbose ułatwiają migrację i ewentualne debugowanie problemów.
Flaga --copy-storage-all, informuje że należy skopiować całą zawartość maszyny. Stanowi ona jedyną różnicę między migracją z dyskiem NFS i bez (w migracji z użyciem dysku NFS flaga ta nie występuje). Następnie podajemy nazwę maszyny wirtualnej oraz ścieżkę do hypervisora na maszynie gdzie zostanie zmigrowany obecny host (w naszym przypadku jest to też adres maszyny na której odpalany jest skrypt). Opcja --migrateuri wymusza port, przez który ma zostać zmigrowana maszyna. Należało dodać taką opcję ponieważ bez niej virsh, nie był w stanie zmigrować VM-ki i zwracał błąd, że ten port jest zablokowany.

Dwie ostatnie komendy (migrate-nfs i migrate-scp) działają w bardzo podobny sposób. Pierwsza z nich migruje wykorzystując dysk NFS, natomiast druga kopiuje wszystkie zasoby. Migracja NFS przeszukuje wszystkich hostów zapisanych w pliku *config.json*, logując się do nich poprzez ssh i sprawdzając czy działa tam poszukiwana maszyna. Jeśli tak to wyłącza ją,

a następnie definiuje (jak jest taka potrzeba) i uruchamia na obecnym hoście (tym, który wywołuje skrypt). Funkcja wykonująca migrację maszyny z użyciem dysku NFS wygląda następująco:

```python
def run_vm_nfs(img_name: str):
    log_info_before(vm_name=vm_name)
    mount_path = get_field_from_config("nfs_path")
    full_image_path = Path(mount_path) / img_name

    if not full_image_path.exists():
        raise FileNotFoundError(f"Image not found: {full_image_path}")

    vm_name = full_image_path.stem
    full_image_path = str(full_image_path)
    local_ip = get_local_ip()

    host_ips = get_field_from_config("client_ips")
    for host_ip in host_ips:
        if host_ip == local_ip:
            logging.info(f"Skipping local host: {host_ip}")
            continue
        logging.info(f"Checking if {img_name} is in use on {host_ip}...")
        image_in_use_result = VMStatus.ERROR_RETRY
        while image_in_use_result == VMStatus.ERROR_RETRY:
            ssh_user = input(f"Enter SSH username for {host_ip}: ").strip()
            password = getpass.getpass(f"Sudo password for {ssh_user}@{host_ip}: ")
            image_in_use_result = remote_image_in_use(host_ip, ssh_user, full_image_path, password)
            if image_in_use_result == VMStatus.STILL_RUNNING:
                logging.info(f"Image {img_name} is in use on {host_ip}")
                if shutdown_remote_vm(host_ip, ssh_user, full_image_path, password) == VMStatus.ERROR_RETRY:
                    raise RuntimeError("Couldn't shut down remote VM.")

                xml_path = copy_vm_xml_config(host_ip=host_ip, vm_name=vm_name, ssh_user=ssh_user, password=password)
                define_vm(vm_name=vm_name, xml_path=xml_path)
                break

    start_vm(vm_name=vm_name)
```

Druga metoda ma podany adres hosta, z którego należy zmigrować maszynę. Kopiuje konfigurację i cały dysk maszyny używając komendy scp. Skrypt wygląda następująco:

```python
def run_vm_scp(host_ip: str, img_name: str):
    vm_name = img_name.rsplit(".", 1)[0]
    log_info_before(vm_name=vm_name, host_ip=host_ip)
    local_dest_dir = Path(get_field_from_config("local_vm_path"))

    local_dest_dir.mkdir(parents=True, exist_ok=True)
    local_dest = local_dest_dir / img_name

    ssh_user = input(f"Enter SSH username for {host_ip}: ").strip()
    password = getpass.getpass(f"Sudo password for {ssh_user}@{host_ip}: ")

    remote_path = look_for_vm_image(host_ip=host_ip, img_name=img_name, ssh_user=ssh_user, password=password)

    image_in_use_result = remote_image_in_use(host_ip, ssh_user, remote_path, password)      zgodek, 2 weeks a
    if image_in_use_result == VMStatus.STILL_RUNNING:
        logging.info(f"Image {img_name} is in use on {host_ip}")
        if shutdown_remote_vm(host_ip, ssh_user, remote_path, password) == VMStatus.ERROR_RETRY:
            raise RuntimeError("Couldn't shut down remote VM.")
    logging.info(f"Starting SCP of image {remote_path}...")

    vm_scp_command = [
        "sshpass", "-p", password,
        "scp",
        f"{ssh_user}@{host_ip}:{remote_path}",
        str(local_dest)
    ]
    vm_scp_result = subprocess.run(vm_scp_command, input=password + "\n", capture_output=True, text=True)

    if vm_scp_result.returncode == 0:
        logging.info(f"VM image copied to {local_dest}")
    else:
        raise RuntimeError(f"Failed to copy VM image:\n{vm_scp_result.stderr}")


    xml_path = copy_vm_xml_config(host_ip=host_ip, vm_name=vm_name, ssh_user=ssh_user, password=password)

    define_vm(vm_name=vm_name, xml_path=xml_path)

    start_vm(vm_name=vm_name)
```

Poniżej załączono zrzuty ekranów podczas wywołania wszystkich komend migracji. Maszyny były wówczas przełączane z pc 192.168.100.82 na laptopa 192.168.100.31, więc skrypt wywoływano na laptopie.

- migrate-local-live

```
zgodek@zg-laptop:~/WSO$ python3 main.py migrate-local-live server-02 192.168.100.82
2025-06-02 12:02:10,569 [INFO] Before attempting to migrate server-02 from 192.168.100.82 please make sure ssh connection is
 enabled on this machine and firewall is properly configured.
2025-06-02 12:02:10,569 [INFO] Please make sure that remote ssh key from 192.168.100.82 is added to your local machine.
Running command: ping -w 1 192.168.100.82
Enter SSH username for 192.168.100.82: zgodek
Sudo password for zgodek@192.168.100.82:
2025-06-02 12:02:51,763 [INFO] Machine migrated successfully.
CompletedProcess(args=['ssh', 'zgodek@192.168.100.82', 'sudo', '-S', 'virsh', 'migrate', '--live', '--persistent', '--unsafe
', '--verbose', '--copy-storage-all', 'server-02', 'qemu+ssh://zgodek@192.168.100.31/system', '--migrateuri', 'tcp://zgodek@
192.168.100.31:49153'], returncode=0, stdout='\n', stderr='[sudo] password for zgodek: \nMigration: [  0 %]\nMigration: [  4
 %]\nMigration: [  5 %]\nMigration: [ 17 %]\nMigration: [ 19 %]\nMigration: [ 20 %]\nMigration: [ 22 %]\nMigration: [ 23 %]\
nMigration: [ 24 %]\nMigration: [ 24 %]\nMigration: [ 25 %]\nMigration: [ 25 %]\nMigration: [ 26 %]\nMigration: [ 30 %]\nMig
ration: [ 31 %]\nMigration: [ 31 %]\nMigration: [ 32 %]\nMigration: [ 33 %]\nMigration: [ 34 %]\nMigration: [ 34 %]\nMigrati
on: [ 35 %]\nMigration: [ 36 %]\nMigration: [ 37 %]\nMigration: [ 37 %]\nMigration: [ 39 %]\nMigration: [ 40 %]\nMigration:
[ 40 %]\nMigration: [ 41 %]\nMigration: [ 42 %]\nMigration: [ 43 %]\nMigration: [ 44 %]\nMigration: [ 44 %]\nMigration: [ 45
 %]\nMigration: [ 46 %]\nMigration: [ 46 %]\nMigration: [ 47 %]\nMigration: [ 47 %]\nMigration: [ 48 %]\nMigration: [ 48 %]\
nMigration: [ 49 %]\nMigration: [ 50 %]\nMigration: [ 58 %]\nMigration: [ 59 %]\nMigration: [ 59 %]\nMigration: [ 69 %]\nMig
ration: [ 71 %]\nMigration: [ 77 %]\nMigration: [ 79 %]\nMigration: [ 79 %]\nMigration: [ 79 %]\nMigration: [ 84 %]\nMigrati
on: [ 86 %]\nMigration: [ 87 %]\nMigration: [ 88 %]\nMigration: [ 93 %]\nMigration: [ 90 %]\nMigration: [ 94 %]\nMigration:
[ 95 %]\nMigration: [ 95 %]\nMigration: [ 96 %]\nMigration: [ 96 %]\nMigration: [ 97 %]\nMigration: [ 97 %]\nMigration: [ 98
 %]\nMigration: [ 98 %]\nMigration: [ 99 %]\nMigration: [100 %]\nMigration: [100 %]')
```

weryfikacja działania na maszyny na laptopie (przed, w trakcie i po migracji):

```
zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 -    server-01   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 11   server-02   paused
 -    server-01   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 11   server-02   running
 -    server-01   shut off
```

- migrate-nfs-live

```
zgodek@zg-laptop:~/WSO$ python3 main.py migrate-nfs-live server-01 192.168.100.82
2025-06-02 11:56:42,290 [INFO] Before attempting to migrate server-01 from 192.168.100.82 please make sure ssh connection is
 enabled on this machine and firewall is properly configured.
2025-06-02 11:56:42,290 [INFO] Please make sure that remote ssh key from 192.168.100.82 is added to your local machine.
Running command: ping -w 1 192.168.100.82
Enter SSH username for 192.168.100.82: zgodek
Sudo password for zgodek@192.168.100.82:
2025-06-02 11:56:57,134 [INFO] Machine migrated successfully.
CompletedProcess(args=['ssh', 'zgodek@192.168.100.82', 'sudo', '-S', 'virsh', 'migrate', '--live', '--persistent', '--unsafe
', '--verbose', 'server-01', 'qemu+ssh://zgodek@192.168.100.31/system', '--migrateuri', 'tcp://zgodek@192.168.100.31:49152']
, returncode=0, stdout='\n', stderr='[sudo] password for zgodek: \nMigration: [  1 %]\nMigration: [ 10 %]\nMigration: [ 36 %
]\nMigration: [ 42 %]\nMigration: [ 47 %]\nMigration: [ 53 %]\nMigration: [ 58 %]\nMigration: [ 64 %]\nMigration: [ 70 %]\nM
igration: [ 76 %]\nMigration: [ 82 %]\nMigration: [ 89 %]\nMigration: [ 95 %]\nMigration: [100 %]')
```

weryfikacja działania maszyny na laptopie:

```
zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 -    server-01   shut off
 -    server-02   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 9    server-01   paused
 -    server-02   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 9    server-01   running
 -    server-02   shut off
```

- migrate-nfs

```
zgodek@zg-laptop:~/WSO$ python3 main.py migrate-nfs server-01.img
2025-06-01 18:03:25,359 [INFO] Skipping local host: 192.168.100.31
2025-06-01 18:03:25,359 [INFO] Checking if server-01.img is in use on 192.168.100.82...
Enter SSH username for 192.168.100.82: zgodek
Sudo password for zgodek@192.168.100.82:
2025-06-01 18:03:28,180 [INFO] Checking if image: /mnt/nfsshare/server-01.img is in use on host: 192.168.100.82.
..
2025-06-01 18:03:28,583 [INFO] Image server-01.img is in use on 192.168.100.82
2025-06-01 18:03:28,583 [INFO] Attempting to shut down VM 'server-01' on host 192.168.100.82 via SSH...
2025-06-01 18:03:28,880 [INFO] Waiting for VM to shut down...
2025-06-01 18:03:33,884 [INFO] Checking if image: /mnt/nfsshare/server-01.img is in use on host: 192.168.100.82.
..
2025-06-01 18:03:34,315 [INFO] Still in use... waiting
2025-06-01 18:03:39,319 [INFO] Checking if image: /mnt/nfsshare/server-01.img is in use on host: 192.168.100.82.
..
2025-06-01 18:03:39,696 [INFO] Still in use... waiting
2025-06-01 18:03:44,701 [INFO] Checking if image: /mnt/nfsshare/server-01.img is in use on host: 192.168.100.82.
..
2025-06-01 18:03:45,085 [INFO] Image is not in use on any remote hosts.
2025-06-01 18:03:45,085 [INFO] Fetching VM xml config from 192.168.100.82...
2025-06-01 18:03:45,562 [INFO] VM config created and copied to /tmp/server-01.xml
2025-06-01 18:03:45,735 [INFO] Changes detected in VM xml for server-01, proceeding with redefine.
2025-06-01 18:03:45,886 [INFO] Successfully defined server-01 locally.
2025-06-01 18:03:45,886 [INFO] Starting VM locally...
2025-06-01 18:03:47,376 [INFO] VM 'server-01' started successfully.
```

weryfikacja działania maszyny na laptopie:

```
zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 -    server-01   shut off
 -    server-02   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 3    server-01   running
 -    server-02   shut off
```

- migrate-scp

```
zgodek@zg-laptop:~/WSO$ python3 main.py migrate-scp 192.168.100.82 server-02.img
Enter SSH username for 192.168.100.82: zgodek
Sudo password for zgodek@192.168.100.82:
2025-06-01 18:08:31,965 [INFO] Searching for server-02.img on remote host 192.168.100.82...
2025-06-01 18:08:49,993 [INFO] Checking if image: /home/zgodek/vms/server-02.img is in use on host: 192.168.100.
82...
2025-06-01 18:08:50,367 [INFO] Image server-02.img is in use on 192.168.100.82
2025-06-01 18:08:50,367 [INFO] Attempting to shut down VM 'server-02' on host 192.168.100.82 via SSH...
2025-06-01 18:08:50,653 [INFO] Waiting for VM to shut down...
2025-06-01 18:08:55,659 [INFO] Checking if image: /home/zgodek/vms/server-02.img is in use on host: 192.168.100.
82...
2025-06-01 18:08:56,044 [INFO] Image is not in use on any remote hosts.
2025-06-01 18:08:56,044 [INFO] Starting SCP of image /home/zgodek/vms/server-02.img...
2025-06-01 18:10:27,932 [INFO] VM image copied to /home/zgodek/vms/server-02.img
2025-06-01 18:10:27,933 [INFO] Fetching VM xml config from 192.168.100.82...
2025-06-01 18:10:28,455 [INFO] VM config created and copied to /tmp/server-02.xml
2025-06-01 18:10:28,659 [INFO] Successfully defined server-02 locally.
2025-06-01 18:10:28,659 [INFO] Starting VM locally...
2025-06-01 18:10:29,905 [INFO] VM 'server-02' started successfully.
```

weryfikacja działania maszyny na laptopie:

```
zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 -    server-01   shut off

zgodek@zg-laptop:~/WSO$ virsh list --all
 Id   Name        State
----------------------------
 4    server-02   running
 -    server-01   shut off
```

8 Ustawienia środowiska do testów z pomiarem czasu

Testy były przeprowadzane przy użyciu Raspberry Pi 3B+ jako serwera NFS. Komputer stacjonarny i laptop pełnił rolę hostów maszyn wirtualnych. Wszystkie urządzenia były w tej samej sieci lokalnej i ustawiono im statyczne adresy IP na ruterze.
Dodatkowo należało umożliwić maszynom wirtualnym komunikację z siecią lokalną, do której podłączony był ich host. Było to potrzebne, żeby móc zbadać dostępność maszyny komendą ping w trakcie jej migracji. Aby to osiągnąć należy stworzyć most sieciowy w libvirt poniższymi trzema komendami:

virsh net-define br_vm.xml
virsh net-start br_vm
virsh net-autostart br_vm

Następnie należy edytować konfigurację sieciową hosta znajdującą się w pliku /etc/network/interfaces. Należy dodać poniższe linijki (pamiętając o takiej samej nazwie mostu sieciowego co w poprzednich komendach)

auto br_vm
iface br_vm inet dhcp

```
    bridge_ports eno1
    bridge_stp on
    bridge_fd 0

iface eno1 inet manual
```

Następnie trzeba aktywować powyższe interfejsy poniższymi komendami:

```
sudo ip link set eno1 up
sudo ip link set eno1 master br_vm
sudo ip link set br_vm up
sudo dhclient br_vm
```

Potem do konfiguracji maszyny (komenda virsh edit <nazwa_maszyny>) należy dodać interfejs z mostem sieciowy wklejając poniższe linijki

```
<interface type='bridge'>
  <mac address='52:54:00:ab:cd:ef'/>
  <source bridge='br_vm'/>
  <model type='virtio'/>
</interface>
```

Po takiej konfiguracji należało ustawić statyczny adres IP w ruterze dla adresu MAC interfejsu sieciowego maszyny wirtualnej. Na zdjęciu widoczna jest konfiguracja rutera dla interfejsów dwóch maszyn, które wykorzystywano do testów (adres IP 192.168.100.25 był przypisany do maszyny znajdującej się w systemie plików NFS – server-01, a 192.168.100.97 należał do maszyny przechowywanej lokalnie na komputerze – server-02):

☐	52:54:00:ab:cd:ef	192.168.100.25
☐	52:54:00:98:c1:46	192.168.100.97

Następnie w maszynie wirtualnej należało zapisać konfigurację interfejsu w pliku /etc/netplan/01-netcfg.yaml

```
network:
  version: 2
  ethernets:
    enp8s0:
      dhcp4: true
```

A także trzeba było zaaplikować zmiany poprzez komendę:

sudo netplan apply

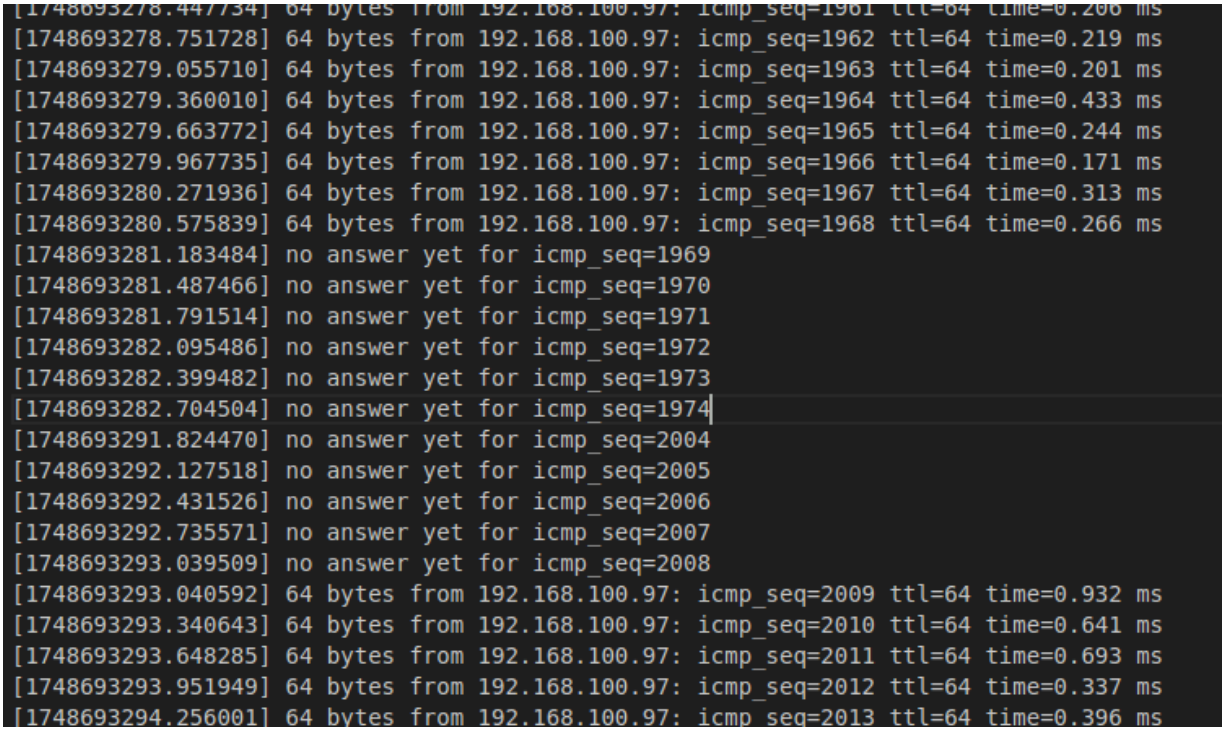
Wówczas maszyna dostawała stały adres IP, który nie zmieniał się w trakcie migracji.

9 Wyniki testów z pomiarem czasu

Testy przeprowadzone zostały z użyciem komendy ping, sprawdzającej przez jaki czas dana maszyna nie opowiada. Dokładna składnia komendy prezentuje się następująco:

ping <ip_addr>-i 0.3 -D -O

Podczas wywołania komendy virsh migrate live wraz z kopiowaniem całej pamięci (migrate-local-live) z lokalnego dysku maszyna zaczęła ponownie odpowiadać po 12,5 sekundach. Poniżej widnieje odczyt komendy ping (po wycięciu części pakietów które nie dotarły). Co ciekawe w tym wypadku czas odpowiedzi się nie zwiększył po rozpoczęciu kopiowania.



Korzystając z migracji przy użyciu komendy virsh migrate live i systemu NFS (migrate-nfs-live) w najlepszym przypadku maszyna zupełnie nie odpowiadała przez 0,6 sekundy. Ale po rozpoczęciu procesu migracji jej czas odpowiedzi się wydłużył z około 0,2 ms do ponad 1 ms. Poniżej widnieje odczyt komendy ping w trakcie migracji:

```
[1748692083.808345] 64 bytes from 192.168.100.25: icmp_seq=52 ttl=64 time=0.758 ms
[1748692084.112291] 64 bytes from 192.168.100.25: icmp_seq=53 ttl=64 time=0.715 ms
[1748692084.416530] 64 bytes from 192.168.100.25: icmp_seq=54 ttl=64 time=0.927 ms
[1748692084.717070] 64 bytes from 192.168.100.25: icmp_seq=55 ttl=64 time=1.08 ms
[1748692085.017573] 64 bytes from 192.168.100.25: icmp_seq=56 ttl=64 time=1.02 ms
[1748692085.318379] 64 bytes from 192.168.100.25: icmp_seq=57 ttl=64 time=1.39 ms
[1748692085.618869] 64 bytes from 192.168.100.25: icmp_seq=58 ttl=64 time=1.27 ms
[1748692085.919514] 64 bytes from 192.168.100.25: icmp_seq=59 ttl=64 time=1.37 ms
[1748692086.220979] 64 bytes from 192.168.100.25: icmp_seq=60 ttl=64 time=1.96 ms
[1748692086.521000] 64 bytes from 192.168.100.25: icmp_seq=61 ttl=64 time=1.52 ms
[1748692086.822101] 64 bytes from 192.168.100.25: icmp_seq=62 ttl=64 time=1.87 ms
[1748692087.123020] 64 bytes from 192.168.100.25: icmp_seq=63 ttl=64 time=1.52 ms
[1748692087.424171] 64 bytes from 192.168.100.25: icmp_seq=64 ttl=64 time=1.74 ms
[1748692087.724671] 64 bytes from 192.168.100.25: icmp_seq=65 ttl=64 time=1.14 ms
[1748692088.025187] 64 bytes from 192.168.100.25: icmp_seq=66 ttl=64 time=1.12 ms
[1748692088.325723] 64 bytes from 192.168.100.25: icmp_seq=67 ttl=64 time=1.12 ms
[1748692088.626205] 64 bytes from 192.168.100.25: icmp_seq=68 ttl=64 time=1.06 ms
[1748692088.926798] 64 bytes from 192.168.100.25: icmp_seq=69 ttl=64 time=1.15 ms
[1748692089.227341] 64 bytes from 192.168.100.25: icmp_seq=70 ttl=64 time=0.999 ms
[1748692089.527940] 64 bytes from 192.168.100.25: icmp_seq=71 ttl=64 time=1.25 ms
[1748692089.828402] 64 bytes from 192.168.100.25: icmp_seq=72 ttl=64 time=0.981 ms
[1748692090.128867] 64 bytes from 192.168.100.25: icmp_seq=73 ttl=64 time=1.32 ms
[1748692090.429423] 64 bytes from 192.168.100.25: icmp_seq=74 ttl=64 time=1.18 ms
[1748692090.729944] 64 bytes from 192.168.100.25: icmp_seq=75 ttl=64 time=1.39 ms
[1748692091.030409] 64 bytes from 192.168.100.25: icmp_seq=76 ttl=64 time=1.10 ms
[1748692091.631487] no answer yet for icmp_seq=77
[1748692091.631887] 64 bytes from 192.168.100.25: icmp_seq=78 ttl=64 time=0.297 ms
[1748692091.935808] 64 bytes from 192.168.100.25: icmp_seq=79 ttl=64 time=0.242 ms
[1748692092.240097] 64 bytes from 192.168.100.25: icmp_seq=80 ttl=64 time=0.453 ms
[1748692092.543949] 64 bytes from 192.168.100.25: icmp_seq=81 ttl=64 time=0.362 ms
[1748692092.847888] 64 bytes from 192.168.100.25: icmp_seq=82 ttl=64 time=0.310 ms
[1748692093.151830] 64 bytes from 192.168.100.25: icmp_seq=83 ttl=64 time=0.273 ms
```

Wykorzystując system NFS oraz ponowne uruchomienie maszyny (migrate-nfs) uzyskaliśmy czas w którym maszyna była nieaktywna równy 26,1 sekund. Wyniki komendy ping widnieją poniżej:

```
[1748694649.416312] 64 bytes from 192.168.100.25: icmp_seq=60 ttl=64 time=0.691 ms
[1748694649.721275] 64 bytes from 192.168.100.25: icmp_seq=61 ttl=64 time=1.31 ms
[1748694650.021211] 64 bytes from 192.168.100.25: icmp_seq=62 ttl=64 time=0.462 ms
[1748694650.631499] no answer yet for icmp_seq=63
[1748694650.935492] no answer yet for icmp_seq=64
[1748694651.239504] no answer yet for icmp_seq=65
[1748694651.543520] no answer yet for icmp_seq=66
[1748694651.847567] no answer yet for icmp_seq=67
[1748694652.151504] no answer yet for icmp_seq=68
[1748694675.255670] no answer yet for icmp_seq=144
[1748694675.559501] no answer yet for icmp_seq=145
[1748694675.863477] no answer yet for icmp_seq=146
[1748694676.167478] no answer yet for icmp_seq=147
[1748694676.167885] 64 bytes from 192.168.100.25: icmp_seq=148 ttl=64 time=0.330 ms
[1748694676.471709] 64 bytes from 192.168.100.25: icmp_seq=149 ttl=64 time=0.168 ms
[1748694676.775644] 64 bytes from 192.168.100.25: icmp_seq=150 ttl=64 time=0.119 ms
[1748694677.079652] 64 bytes from 192.168.100.25: icmp_seq=151 ttl=64 time=0.139 ms
```

Wykorzystując ostatnią komendę kopiującą pamięć (o wielkości 11 GB) z użyciem scp (migrate-scp), uzyskaliśmy okres wyłączenia równy 108,3 sekundy. Poniżej widnieją wyniki komendy ping:

```
[1748695892.968548] 64 bytes from 192.168.100.97: icmp_seq=101 ttl=64 time=0.968 ms
[1748695893.268860] 64 bytes from 192.168.100.97: icmp_seq=102 ttl=64 time=0.840 ms
[1748695893.576798] 64 bytes from 192.168.100.97: icmp_seq=103 ttl=64 time=1.03 ms
[1748695893.877732] 64 bytes from 192.168.100.97: icmp_seq=104 ttl=64 time=1.21 ms
[1748695894.178135] 64 bytes from 192.168.100.97: icmp_seq=105 ttl=64 time=0.944 ms
[1748695894.783513] no answer yet for icmp_seq=106
[1748695895.087705] no answer yet for icmp_seq=107
[1748695895.391493] no answer yet for icmp_seq=108
[1748695895.695525] no answer yet for icmp_seq=109
[1748695895.999509] no answer yet for icmp_seq=110
[1748695896.303513] no answer yet for icmp_seq=111
[1748695896.607761] no answer yet for icmp_seq=112
[1748695896.911512] no answer yet for icmp_seq=113
[1748695897.215506] no answer yet for icmp_seq=114
[1748696002.503604] From 192.168.100.31 icmp_seq=325 Destination Host Unreachable
[1748696002.503637] From 192.168.100.31 icmp_seq=326 Destination Host Unreachable
[1748696002.503649] From 192.168.100.31 icmp_seq=327 Destination Host Unreachable
[1748696002.503660] From 192.168.100.31 icmp_seq=328 Destination Host Unreachable
[1748696002.503670] From 192.168.100.31 icmp_seq=329 Destination Host Unreachable
[1748696002.503680] From 192.168.100.31 icmp_seq=330 Destination Host Unreachable
[1748696002.503686] no answer yet for icmp_seq=330
[1748696002.504008] 64 bytes from 192.168.100.97: icmp_seq=331 ttl=64 time=0.283 ms
[1748696002.807745] 64 bytes from 192.168.100.97: icmp_seq=332 ttl=64 time=0.197 ms
[1748696003.111791] 64 bytes from 192.168.100.97: icmp_seq=333 ttl=64 time=0.229 ms
[1748696003.415725] 64 bytes from 192.168.100.97: icmp_seq=334 ttl=64 time=0.184 ms
[1748696003.719680] 64 bytes from 192.168.100.97: icmp_seq=335 ttl=64 time=0.112 ms
```

10 Wnioski

Migracja maszyn wirtualnych okazała się nie być trudnym zagadnieniem. Szczególnie prosta była migracja live przy użyciu programu virsh. Natomiast migracja z zatrzymywaniem maszyny przez ssh była wolniejsza, bardziej złożona do zaprogramowania, a także powodowała kilkukrotnie większe przestoje w działaniu maszyny niż migracja live. Ale problemem w migracji live była mała stabilność, którą zauważyliśmy przy przenoszeniu maszyny z kopiowaniem dysku. W rzeczywistym przypadku najlepsza prawdopodobnie byłaby migracja live wraz z połączeniem z rozproszonym systemem plików. Oferuje najmniejszy czas przestoju i wymaga wywołania małej ilości komend (szczególnie jeżeli wykonywałoby się to ręcznie, a nie naszym programem).